2-2010

Paying Attention to What Matters: Observation Abstraction in Partially Observable Environments

Alicia Peregrin Wolfe
University of Massachusetts Amherst, pippin.wolfe@gmail.com

PAYING ATTENTION TO WHAT MATTERS: OBSERVATION ABSTRACTION IN PARTIALLY OBSERVABLE ENVIRONMENTS

A Dissertation Presented

by

ALICIA PEREGRIN WOLFE

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2010

Computer Science

PAYING ATTENTION TO WHAT MATTERS: OBSERVATION ABSTRACTION IN PARTIALLY OBSERVABLE ENVIRONMENTS

A Dissertation Presented

by

ALICIA PEREGRIN WOLFE

Approved as to style and content by:

_____

Andrew G. Barto, Chair

_____

Sridhar Mahadevan, Member

_____

Shlomo Zilberstein, Member

_____

Leslie Kaelbling, Member

_____

Bruce Turkington, Member

_____

Andrew G. Barto, Department Chair
Computer Science

To my mother, Mary Anne Schweitzer, for her time and patience.

ACKNOWLEDGMENTS

Thanks firstly to my committee, in particular for bearing with me through several schedule changes. Also to the members of the Autonomous Learning Laboratory for many interesting discussions, including but not limited to Ozgur Simsek, Amy McGovern, Balaraman Ravindran, Sarah Osentoski and Ashvin Shah. Other members of the UMass Computer Science community I've enjoyed many long discussions with include Victoria Manfredi, Jen Neville, Lisa Friedland, Emily Horrell and TJ Brunette. Prof. David Jensen, while not on the committee for my dissertation, was a helpful mentor and collaborator on earlier projects.

Supportive friends and family include: Martin Walkow, providing the linguist's perspective; my sister Rachel Wolfe who can always make me see the humor in any situation; my father John Wolfe; who taught me to always question, question, question; and my mother Mary Anne Schweitzer, who, in addition to probably hundreds of long phone calls pitched in at the last minute to transport my shoes into town from Connecticut.

Also thanks to the many helpful staff members in the department, including but not limited to Leeanne Leclerc, Barb Sutherland and Gwyn Mitchell.

ABSTRACT

PAYING ATTENTION TO WHAT MATTERS: OBSERVATION ABSTRACTION IN PARTIALLY OBSERVABLE ENVIRONMENTS

FEBRUARY 2010

ALICIA PEREGRIN WOLFE

Combined B.A./B.Sc., BROWN UNIVERSITY

M.Sc., UNIVERSITY OF MASSACHUSETTS, AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Andrew G. Barto

Autonomous agents may not have access to complete information about the state of the environment. For example, a robot soccer player may only be able to estimate the locations of other players not in the scope of its sensors. However, even though all the information needed for ideal decision making cannot be sensed, all that is sensed is usually not needed. The noise and motion of spectators, for example, can be ignored in order to focus on the game field. Standard formulations do not consider this situation, assuming that all the can be sensed must be included in any useful abstraction.

This dissertation extends the Markov Decision Process Homomorphism framework (Ravindran, 2004) to partially observable domains, focusing specically on reducing Partially Observable Markov Decision Processes (POMDPs) when the model is known. This involves ignoring aspects of the observation function which are irrelevant to a particular

task. Abstraction is particularly important in partially observable domains, as it enables the formation of a smaller domain model and thus more efficient use of the observed features.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

One of the most important tasks an independent learning agent faces is to separate what is important and relevant from what is not — to separate the wheat from the chaff. Focusing on essential details can make a task possible: for example, when driving it is not advised that one take in all of the scenery — focusing on the road and road signs improves performance on the task (and safety). Separating what is relevant from what is not can turn an intractable learning problem into a solvable one by reducing the complexity of the problem. In humans, this is most obvious in competitive situations: the chess player whose focus on the board is absolute, the basketball player who ignores the fans in the bleachers to focus only on the basket. This is even more important when the state is only partially observable.

The agent must solve two problems in order to construct a useful representation for a task when state is partially observable. First, some information may be missing, and must be inferred from the available observations: for example, the location of cars on the road behind a driver, or hidden by other cars ahead of the driver. Typically it is assumed that all of the information that is observed will be useful in making inferences about the missing information. For example, the behavior of the other visible cars on the road can alert the driver to hidden obstacles ahead. However, not all information is necessarily useful: paying attention to details of the scenery, or cloud patterns overhead, is likely to distract the driver, making the task more difficult. The second problem an agent must solve in order to find a good representation is the problem of deciding which information to ignore.

Existing literature on abstraction for learning and planning in both partially and fully observable problems can largely be grouped into two catagories:

- Specific task abstraction methods
- General purpose abstract representations

One example of the task-specific abstraction approach is the the UTree algorithm (McCallum, 1995). UTree is a decision-tree based abstraction algorithm, in which relevant features are chosen to fit a specific task. Methods like value-directed compression for PSRs (Poupart and Boutilier, 2002) and DEC-POMDPs (Carlin and Zilberstein, 2008) also fall in this category. The advantage of taking a task-specific approach is that the abstract model can ignore as much information as possible, leading to faster learning and planning.

Methods like Proto-Value Functions (PVFs) (Mahadevan, 2005) and Action Respecting Embeddings (AREs) (Bowling et al., 2005) fall at the other end of this spectrum. [1] They create general purpose abstractions by transforming the agent's representation of the environment to more closely mimic the structure of that environment. In the case of PVFs, this structure is the graph structure of the transitions between states of the environment, while in the case of AREs, the structure is the local action transition behavior. Methods from this catagory have two advantages: first, the abstract models may be constructed before the agent knows what task it will need to perform, and second, the abstract models may be used for multiple tasks. However, this generality means that in some cases the models may be larger than a task-specific model would be, resulting in lower savings when planning or learning.

This dissertation chiefly addresses an intermediate type of abstraction, one that falls between the single task and general purpose abstract representations (see Figure 1.1). This type of abstraction is built to predict a specific aspect of the environment, represented via an output function. The output function could be anything from a boolean variable indicating

[1] Neither of these examples have yet been adapted to accommodate partial observability.

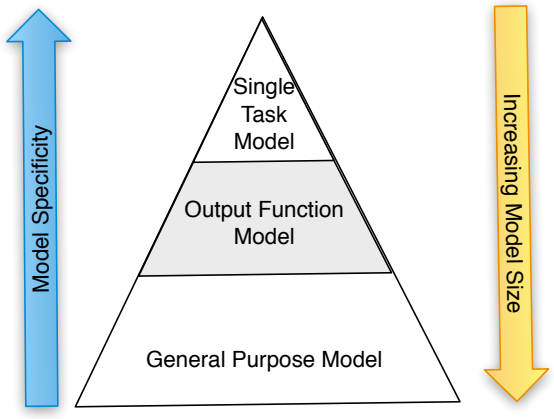


Figure 1.1. Classes of abstraction methods, ordered in terms of both the specificity of the models they create (from general purpose models to task specific models) and the size of the abstract models they typically create. General purpose models must generally include more information than task specific models, while output function models occupy a middle ground between the two extremes.

whether some test is currently true ("is the glass on the table?") to a feature of a specific object in the agent's environment ("how much water is in the glass?"). An abstract output-function model must support planning and learning for any task that depends only on the output function. In the first example, supported tasks could achieve each possible setting: glass on or off the table. In the second example, any task that controls the level of water in the glass (fill the glass, empty the glass, etc.) would be supported.

An abstract model is only useful if the savings garnered from using the model outweigh the cost of constructing it. Unlike the single task learning models, output-drected models are not single-use abstractions, to be discarded once their target policy has been constructed. In Wolfe and Barto (2006) we show that if the output function is chosen wisely, an output-directed abstract model can be reused for multiple related tasks, amortizing the cost of constructing the abstract model over multiple uses. However, learning and planning only become more efficient if there is information that can be ignored by the abstract model. Focusing on a single output function, rather than every possible task, can also allow more information to be ignored than in general purpose abstract representations.

Algorithms like UTree can be adapted to build output-directed models. Wolfe and Barto (2006) uses an adapted UTree algorithm to find output-directed abstractions when the state is fully observed, but UTree was originally designed for situations with partially observable state, and thus the same algorithm could be applied to partially observable situations. This dissertation does not take quite that direction. UTree is a fast, approximate algorithm, with several drawbacks, relating both to the fact that it is based on a Decision Tree algorithm, and the fact that features are constructed over the entire history of the agent. Holmes and Isbell (2006) address some of the issues with history-based features.

However, rather than focussing on a specific approximation algorithm, this dissertation examines a more idealized case, in which:

- an accurate model is given
- an exact (accurate) abstraction of this model is required
- the observations have not been factored into features

It is unlikely that there are many cases in which an exact solution of this nature would be practical. In practice, some approximate algorithm is likely to be the best option when an agent's computational time is limited. Nonetheless, the abstraction definitions and algorithms that will be presented here serve several purposes. First, they will demonstrate that polynomial time algorithms for abstraction construction are possible, even in this most strict case (although it may not always be possible to find the smallest possible abstract model in polynomial time). This is a step forward, as existing exact methods are exponential time in the worst case. Approximation methods will presumably further improve on the run time. Second, when forming approximate abstract models it is important to understand what is being approximated: a good approximation is not as useful if it is an approximation of a poor target. Since the algorithms in this dissertation all find accurate models, the main trade off the speed of the algorithm and the quality in terms of size of the abstract models it finds. Each of the algorithms outlined in this dissertation includes two components: an ac-

ceptance criteria for abstract models, and a search strategy for finding an acceptable model. Stricter acceptance criteria generally allow for faster evaluation, but may reject the smallest abstract models in some cases. Several acceptance criteria and search strategies will be evaluated relative to one another according to their speed and abstract model acceptance sets.

1.1 Background: Model Minimization

This section covers definitions and methods developed for abstraction when the state is fully observed, in particular Model Minimization in Markov Decision Processes. Model Minimization was originally developed as a single-task abstraction approach, however, the abstract models this approach produces are powerful enough to be used as output-directed abstractions. These methods will be adapted throughout the remainder of the dissertation to extend to cases where the state is not fully observed. This section first outlines the single-task fully-observed Model Minimization definition, then reviews the modifications needed to adapt this approach to more general output functions.

A Markov Decision Process (MDP) consists of a tuple (S, A, T, R). S is a set of states and A a set of actions. The transition function $T : S \times A \times S \rightarrow [0, 1]$ represents the probability of transitioning to each possible next state, given the previous state and action. The reward function ($R : S \times A \rightarrow \mathbb{R}$) represents the expected reward the agent receives for being in a particular state and executing an action.

One of the earliest Model Minimization frameworks was based on methods used for deterministic planning with logical propositions (Dearden and Boutilier, 1997). In this type of model, each action is defined as set of logical pre and post conditions. As initially suggested in Nicholson and Kaelbling (1994), each action may have multiple non-overlapping logical pre-conditions, each of which corresponds to a different distribution over post conditions. The preconditions partition the state space into blocks: in each block the same action has the same effect on the post-condition variables. For example, in a gridworld the

action of going forward might change the agent's location when the proposition "in front of a wall" is false, but not when this proposition is true. By examining the pre and post conditions of possible chained sequences of actions, all propositions relevant to a particular reward function can be found in time linear in the the number of actions and number of propositions used to represent the state (Dearden and Boutilier, 1997). Later versions of this work added situation calculus and first-order axioms with objects (Boutilier et al., 2001).

Dean and Givan (1997) and Givan et al. (2003) take a similar approach but base their method explicitly on partitioning the state space according to the principles of stochastic bisimulation, based on work on concurrent processes (Park, 1981), automata theory (Hartmanis and Stearns, 1966) and stochastic processes (Kemeny and Snell, 1960). An initial partition based on the reward function is constructed first. This is then refined by splitting the states into "stable" blocks: blocks in which the prediction of the next block which will be encountered by the agent is uniform over all states in the same block. Givan et al. (2003) also added a notion of "action-equivalence", in which different actions which have the same effect map to the same abstract action. A simple example of this might be the two alternate methods of tying one's shoelaces: both "actions" have the same ultimate effect of creating a bow shape that holds the shoe closed.

Ravindran (2004) moves from the stochastic bisimulation notion to the notion of a mathematical homomorphism between the true MDP and the abstract model (also an MDP), again drawing on literature on automata theory and stochastic processes. A homomorphism, in general, is a mapping, possibly many to one, that preserves some important aspects of the original system. In the case of MDP homomorphisms, this mapping is from the states and actions of an MDP to the states and actions of an abstract MDP, and preserves both the reward function, and the abstract transition function. The reward function is task specific — an agent that must drive would have a different reward function from an agent that must climb trees, for example. By focusing only on state and action distinctions in the

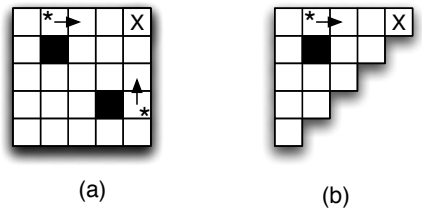


Figure 1.2. Action mappings would enable a reduction (b) of this simple gridworld (a). This is a simpler example of the symmetric gridworld from (Ravindran, 2004). The marked square in the upper right corner is the "goal" and has positive reward, all other states have small negative reward. Both starred states map to the same abstract state, and the actions marked by arrows map to the same abstract action. Note that depending on which actual state the agent is in, the abstract action may correspond to either the action "right" or "up".

domain that are relevant to this specific function, homomorphic reduction can reduce the complexity of learning the task.

An MDP homomorphism (Ravindran, 2004) is a mapping, $h : S \times A \rightarrow \bar{S} \times \bar{A}$, from the states and actions of an MDP $M = (S, A, T, R)$, to an abstract model MDP $\bar{M} = (\bar{S}, \bar{A}, \bar{T}, \bar{R})$. The mapping h must preserve both the reward function and some properties of the transition probabilities of M. Specifically, h consists of a set of mappings: $f : S \rightarrow \bar{S}$, and for each $s \in S$ a mapping $g_s : A \rightarrow A'$ that recodes actions in a possibly state-dependent way. The following properties must hold for all state and action pairs s, a and each next state s':

$$\bar{R}(f(s), g_s(a)) = R(s, a) \tag{1.1}$$

$$P(f(s') \mid f(s), g_s(a)) = \sum_{s'' \in [s']_f} P(s'' \mid s, a). \tag{1.2}$$

where $[s']_f = \{s | f(s) = f(s')\}$.

When a state mapping f can be found that is many-to-one, the abstract MDP $\bar{M}$ has fewer states than M. The gridworld shown in figure 1.2a, for example, can be reduced to the model shown in 1.2b. The homomorphism conditions ensure that $\bar{M}$ accurately tracks the transitions and rewards of M but at the resolution of blocks of states and actions.

A reward function specifies positive or negative feedback for being in certain states and performing certain actions. This can be translated into a policy for achieving optimal reward over time. This optimal policy can be calculated via a *value function* $V^* : S \rightarrow I\!R$. For any state s, the optimal value function $V^*(s)$ is defined as (Sutton and Barto, 1998):

$$V^*(s) = \max_{a \in A} \left[R(s,a) + \gamma \cdot \sum_{s' \in S} P(s' \mid s,a) \cdot V^*(s') \right]. \tag{1.3}$$

where γ is a discount factor between 0 and 1 that causes events further in the future to be given less weight. The optimal policy chooses the action with the highest expected value. The optimal action in state s is:

$$\arg\max_{a \in A} \left[R(s,a) + \gamma \cdot \sum_{s' \in S} P(s' \mid s,a) \cdot V^*(s') \right]$$

Let $\bar{V}^* : \bar{S} \rightarrow I\!R$ be the optimal value function in the abstract MDP $\bar{M}$. It has been shown (Ravindran, 2004) that the abstract and true value functions are the same for any given state s:

$$\bar{V}^*(f(s)) = V^*(s). \tag{1.4}$$

This property guarantees that policies optimal for $\bar{M}$ can be *lifted* to produce optimal policies of the larger MDP M (Ravindran, 2004; Givan et al., 2003). That is, for any state s, the optimal action in s can be calculated by examining the abstract model and the value function for the abstract MDP $\bar{M}$ can be used to produce a policy for the MDP M.

The MDP Homomorphism framework can be used to construct task-specific abstract models for MDPs. The framework lends itself to output function abstraction as well, however, as the next section will show.

1.1.1 Controlled Markov Process Homomorphisms

The MDP homomorphism definition was designed to focus on a single reward function. Contolled Markov Process Homomorphisms Wolfe and Barto (2006) extend the framework to more general output functions.

A Controlled Markov Process (CMP) is an MDP without the latter's reward function: (S, A, T). Reward functions make up one possible subcategory of functions over the states and actions of a CMP. However, other more general classes of functions are also possible. For example, given a set of symbols Y, an output function $\Upsilon : S \times A \times Y \to [0, 1]$ could represent the probability of observing each output symbol after each state and action pair, so that:

$$\Upsilon(s, a, y) = P(y \mid s, a).$$

The same basic principles and algorithms used to define MDP Homomorphisms can be used to create homomorphisms which preserve predictions about Υ.

A CMP Homomorphism is defined as a mapping h from a CMP with output $C = (S, A, T, Y, \Upsilon)$ to an abstract CMP with output $\bar{C} = (\bar{S}, \bar{A}, \bar{T}, Y, \bar{\Upsilon})$. The homomorphism h is again made up of two parts: a state mapping function $f : S \to \bar{S}$ and a state-specific action mapping function $g_s : A \to \bar{A}$. The following constraints must be satisfied, for all states s, s' and actions a:

$$P(y \mid f(s), g_s(a)) = P(y \mid s, a) \tag{1.5}$$

$$P(f(s') \mid f(s), g_s(a)) = P(f(s') \mid s, a), \tag{1.6}$$

where:

$$P(f(s') \mid s, a) = \sum_{s'' \in [s']_f} P(s'' \mid s, a).$$

If these constraints are satisfied predictions and control strategies for Y calculated in $\bar{C}$ can be accurately lifted to C. For any *supported* reward function $r : Y \rightarrow \mathbb{R}$ defined over the output function, the value of a state in C is given by:

$$V^*(s) = \max_{a \in A} \left[\sum_{y \in Y} r(y) \cdot P(y \mid s, a) + \gamma \cdot \sum_{s' \in S} P(s' \mid a, s') \cdot V^*(s') \right],$$

where γ is a discount factor between 0 and 1.

The CMP Homomorphism constraints ensure that:

$$V^*(s) = \bar{V}^*(f(s))$$

and thus, the value function can be calculated using the abstract model in order to find the associated policy in the original model.

1.1.2 Model Minimization in Partially Observable Domains

There are two popular approaches to modeling partial observability. Partially Observable MDPs (POMDPs) (Kaelbling et al., 1998) model use "hidden" state to model the unobservable portions of the state, while Predictive State Representations (PSRs) (Littman et al., 2001) model the hidden aspects of the environment using predictions about future observations.

According to Givan et al. (2003) :

> The simplest way of using model-reduction techniques to solve partially observable MDPs (POMDPs) is to apply the model-minimization algorithm to the underlying fully observable MDP using an initial partition that distinguishes on the basis of *both reward and observation model*. The reduced model can then be solved using a standard POMDP algorithm. (emphasis mine)

This implies that every observation distinction observed by the agent must be modeled, predicted and used by the abstract model. And yet it is often the case that some aspects of the observations should not be included.

The simplest way to adapt the MDP Homomorphism framework for partial observability if abstraction over observations is desired is to transform the partially observable problem into an MDP. This is the approach taken by Soni and Singh (2007) in their work on Predictive State Representations (PSRs). There are three ways to construct a fully observable state set from a partially observable environment. The first possibility is to treat the entire history of actions and observations at any given point as a state. The set of possible histories the agent might encounter is quite large — if there are n possible observations, and m actions the agent might take, there can be up to $(nm)^t$ histories in the set of histories of length t, and there is no upper bound on the length of the history collected.

In a PSR or POMDP, "state" is maintained as a real-valued vector, which serves as a sufficient statistic for history. These vectors can be treated as the states of an MDP. However, the number of reachable real-valued state vectors can again be quite large — in the worst case, the number of reachable state vector differs by a small constant from the number of histories. Applying the CMP Homomorphism or MDP Homomorphism constraints to any of these "meta" MDPs is therefore not a practical approach, though it does represent the ideal homomorphism definition.

Even in this dissertation, the algorithms presented will not perfectly achieve this ideal. The goal of any abstraction algorithm is to find a small abstract model (for example, one in which the size of the abstract state space is minimized), but the algorithm must do so within a reasonable amount of time.

In order to discuss this trade-off between abstract model size and algorithm run time, it will be helpful to discuss two distinct aspects of the search for an abstract model:

1. The acceptance and rejection criteria that distinguishes between accurate and inaccurate abstractions.

2. The search algorithm that generates candidate abstractions.

Applying the MDP or CMP Homomorphism constraints to any of the three meta-MDPs defined above produces perfect acceptance and rejection criteria for abstract models Search algorithms designed to find MDP or CMP Homomorphisms find the minimal abstract mode. However, the time complexity of these search algorithms is polynomial in the size of the state space, which in this case corresponds to the number of reachable histories or state vectors. Therefore, both of these criteria will need to be relaxed to achieve polynomial time abstraction search algorithms.

Talvitie et al. (2008) present a search algorithm with exponential worst case run time This is at least a bounded worst case run time. The algorithm examines pairs of histories in order to test the abstraction for correctness. Rather than examining all histories, however, they show that the length of the histories that must be examined is limited to k, where k is the dimensionality of the state vector. There are no more than $(mn)^k$ histories of this length, and in some cases, the required history length may be quite a bit shorter (although the algorithm cannot detect this). However, the algorithm is not guaranteed to find the smallest possible model. Instead, it identifies a family of accurate abstractions for the desired output function, and uses heuristics to choose among them.

The algorithms presented in the next several chapters also relax the first property, the acceptance and rejection criteria by which abstractions are accepted or discarded by the search algorithm. Each algorithm presented here has the following property: whenever the history-based homomorphism definition would reject a particular model, they reject it, though they accept some subset of the accurate abstract models.

In general, stricter abstraction acceptance criteria imply that smaller abstract models may be rejected in favor of larger abstract models that satisfy the criteria. Figure 1.3 illustrates the abstract model acceptance sets for the history based acceptance criteria, as well as the three types of abstraction criteria that will be presented in this dissertation. Each acceptance criteria is based on and defined in terms of a different type of abstract partially

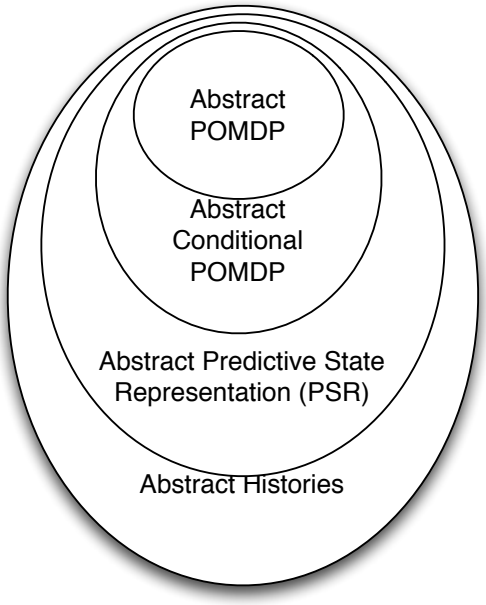


Figure 1.3. The model acceptance sets for the range of algorithms presented in this dissertation.

observable model. The algorithms avoid the exponential running time of the existing algorithms by examining local characteristics of the abstract model. In the abstract POMDP, these local characteristics are abstract state predictions for individual state/action pairs. The abstract conditional POMDP uses similar local characteristics to test each model, though the model itself has a different form. In the case of the abstract PSR, the local characteristics are the probabilities of abstract tests.

In the case of this family of abstract models, the size of the acceptance set is directly related to the worst case running time of the corresponding abstraction search algorithm. The algorithm that searches for an accurate abstract POMDP is faster than both the abstract conditional POMDP search algorithm and the abstract PSR search algorithm. However, it also has the smallest acceptance set, and this may add complexity to the abstract model found.

The following chapters define each of these types of abstract models, along with their associated acceptance and rejection tests and search algorithms. The acceptance set rela-

tionships shown in Figure 1.3 will be proven, and examples of the types of environments that cannot be accurately reduced by each algorithm will be presented.

CHAPTER 2

POMDP HOMOMORPHISMS: POMDP TO POMDP ABSTRACTION

2.1 Introduction

This chapter focuses on the most restrictive abstract model acceptance criteria in this dissertation: the abstract Partially Observable MDP (POMDP) criteria. POMDPs are a widely used model for partial observability. This chapter outlines methods that find a mapping from a POMDP to an abstract POMDP, using the MDP/CMP Homomorphism definitions outlined in the previous chapter as a starting point.

2.2 Partial Observability

In an MDP or CMP, the full model state is observed. A Partially Observable MDP (POMDP), on the other hand, does not include fully observed state. Instead, a set of observations are used to make inferences about the state, which is hidden.

A POMDP (Kaelbling et al., 1998) is defined as tuple (S, A, T, O, Ω), where S, A and T form an underlying CMP. O is the observation set, and $\Omega : S \times A \times O \rightarrow [0, 1]$ is the observation function, which gives the probability of each observation after each state and action: $\Omega(s, a, o) = P(o|s, a)$. Over time, action/observation sequences accumulate into histories. The history set H contains the empty history λ and inductively, for any $h \in H$, $o \in O$, and $a \in A$, H contains hao.

Predictions about the future, planning and learning in a POMDP can be calculated using belief state. A belief state encodes the probability of being in each state, in the form of a history-specific function $b_h : S \rightarrow [0, 1]$. Each element $b_h(s)$ is the probability of being in

state $s \in S$ after observing history h. The initial belief state, b_λ, for the empty history must be specified as an auxiliary portion of the POMDP definition. This belief is updated over time over time using the following formula, for each state s':

$$b_{hao}(s') = \frac{P(o|s', a) \cdot \sum_{s \in S} P(s'|s, a) \cdot b_h(s)}{\sum_{s'' \in S} P(o|s'', a) \cdot \sum_{s \in S} P(s''|s, a) \cdot b_h(s)}. \tag{2.1}$$

This update rule can be separated into two steps, the action update and the observation update:

$$b_{ha}(s') = \sum_{s \in S} P(s'|s, a) \cdot b_h(s) \tag{2.2}$$

$$b_{hao}(s') = \frac{P(o|s', a) \cdot b_{ha}(s')}{\sum_{s'' \in S} P(o|s'', a) \cdot b_{ha}(s'')}. \tag{2.3}$$

For a particular POMDP and initial belief state pairing, some histories can be generated by the transition and observation functions, and some cannot. Let H_M denote the set of valid histories for the POMDP M:

$$H_M = \{h \mid \sum_{s \in S} b_h(s) > 0\}. \tag{2.4}$$

Often multiple histories have the same belief state. The set

$$B_M = \{b_h \mid h \in H_M\} \tag{2.5}$$

contains all unique belief states reachable via some history.

The value function in a POMDP can be defined in terms of belief states. For a reward function $r : S \times A \to \mathbb{R}$:

$$V^*(b_h) = \max_a \left[\sum_{s \in S} r(s, a) \cdot b_h(s) + \gamma \sum_{o \in O} V^*(b_{hao}) \cdot \sum_{s' \in S} P(o \mid s', a) \cdot b_{ha}(s') \right] \tag{2.6}$$

2.3 POMDP Homomorphisms

A POMDP Homomorphism is a mapping from a POMDP $M = (S, A, T, O, \Omega)$ to an abstract POMDP $\bar{M} = (\bar{S}, \bar{A}, \bar{T}, \bar{O}, \bar{\Omega})$. It is made up of three mappings: the state mapping $f : S \rightarrow \bar{S}$, action mapping $g : A \rightarrow \tilde{A}$ and observation mapping $\kappa : O \rightarrow \bar{O}$. The action mapping function g is not state specific, since state specific action mapping functions (the set of functions g_s, with one function per state s) could create conflicts in the policy for some belief states.

The abstract initial belief function $\bar{b}_\lambda$ for $\bar{M}$ is defined in terms of b_λ, the initial belief function for M:

$$\bar{b}_\lambda(f(s)) = \sum_{s' \in [s]_f} b_\lambda(s') \tag{2.7}$$

The set of abstract histories for the abstract POMDP $\bar{M}$ is $\bar{H}$. A history mapping $\chi : H \rightarrow \bar{H}$ from histories of M to abstract histories of $\bar{M}$can be defined using the action map g and observation map κ:

$$\begin{aligned} \chi(\lambda) &= \lambda \\ \chi(hao) &= \chi(h)g(a)\kappa(o). \end{aligned} \tag{2.8}$$

The abstract belief state after history h will be denoted $\bar{b}_{\chi(h)}$.

Soni and Singh (2007) use a similar abstract history mapping function to define homomorphisms for Predictive State Representations (PSRs), which are an alternative representation for partially observable domains, as discussed in Section 1.1.2.

As with CMP Homomorphisms, rather than focusing on a specific reward function, POMDP Homomorphisms are defined with respect to an output function, which might be some feature like position, color, etc. The POMDP output function ζ is defined over observations and output symbols in the output set Z: $\zeta : Z \times O \rightarrow [0, 1]$. The first

constraint that a valid POMDP homomorphism must obey is that the abstract observations must predict the output symbols, by preserving the output function:

$$\zeta(o, z) = \bar{\zeta}(\kappa(o), z) \tag{2.9}$$

This is the first POMDP Homomorphism constraint. [1]

The abstract observation function $\bar{\Omega}$ is defined over abstract states, actions and observations: $\bar{\Omega} : \bar{S} \times \bar{A} \times \bar{O} \rightarrow [0, 1]$. It must also be consistent with the original observation function Ω. This leads to the second POMDP Homomorphism constraint. For all states s, actions a, and observations o:

$$\bar{\Omega}(f(s), g(a), \kappa(o)) = \sum_{o' \in [o]_\kappa} \Omega(s, a, o').$$

Written in probability notation the constraint is:

$$\begin{aligned} P(\kappa(o) \mid f(s), g(a)) &= \sum_{o' \in [o]_\kappa} P(o' \mid s, a) \\ &= P(\kappa(o) \mid s, a). \end{aligned} \tag{2.10}$$

This constraint implies that all states which map to the same abstract state must have the same abstract observation probabilities $P(\kappa(o) \mid s, a)$, for all a, o. Note the similarity between this equation and Equation 1.5, which is the output constraint in a CMP.

The abstract transition function $\bar{T}$ must also be consistent with the original transition function T. This leads to the third POMDP Homomorphism constraint. For all states s, actions a, and next states s':

[1] It is also possible to define the output function over states and actions, rather than observations. We have chosen to use this definition as it simplifies the notation slightly.

$$\bar{T}(f(s), g(a), f(s')) = \sum_{s'' \in [s']_f} T(s, a, s'').$$

written in probability notation this is:

$$\begin{aligned} P(f(s') \mid f(s), g(a)) &= \sum_{s'' \in [s']_f} P(s'' \mid s, a) \\ &= P(f(s') \mid s, a). \end{aligned} \tag{2.11}$$

Note the similarity between this equation and Equation 1.6, which is the transition constraint in a CMP.

Belief state updates in the abstract POMDP proceed according to the definitions of $\bar{T}$ and $\bar{\Omega}$.

The constraints in Equations 2.9 - 2.11 are not sufficient without one additional constraint over the belief states of M and $\bar{M}$, for each history h in H_M:

$$\bar{b}_{\chi(h)}(f(s)) = \sum_{s' \in [s]_f} b_h(s') \tag{2.12}$$

This last constraint is difficult to verify directly, since naively it requires the analysis of every history in H_M. However, several similar but more restrictive constraints can be used in its place. Most of this dissertation will be dedicated to defining alternative constraints which:

1. Can be evaluated in polynomial time in the size of the POMDP state, action and observation sets.

2. Always reject candidate homomorphisms that Equation 2.12 rejects.

3. Accept many of the candidate homomorphisms that Equation 2.12 accepts.

The trade off will be between the speed of the evaluation time and the number of candidate mappings accepted. Rejecting acceptable mappings generally means accepting a possibly larger abstract POMDP which fits more stringent constraints.

The POMDP Homomorphism constraints specified by Equations 2.9 - 2.12 entail a number of useful properties, the most important of which is that the POMDP value function is preserved if they are satisfied.

Before delving into these properties, it is useful to give a few notational details. For any abstract observation $\bar{o} \in \bar{O}$, the label $\bar{o}$ will be used as shorthand for the set of observations which map to the abstract observation $\bar{o}$, particularly in the term $o \in \bar{o}$, which should be read as $o \in \{o|\kappa(o) = \bar{o}\}$. This means, for example, that the statements $\forall o \in \bar{o}, \kappa(o) = \bar{o}$, and $\{o \in \kappa(o')\} = [o']_\kappa$ are both true. Similarly, for an abstract state $\bar{s} \in \bar{S}$, the notation $s \in \bar{s}$ is shorthand for $s \in \{s|f(s) = \bar{s}\}$. The shorthand $b_h(\bar{s})$ refers to $\sum_{s \in \bar{s}} b_h(s)$ for $\bar{s} \in \bar{S}$.

An abstract observation set $\bar{O}$ is *self-sufficient* (Pfeffer, 2001) if it predicts itself — that is, for any history h and action a, the history mapping function χ and action mapping function g must preserve accurate predictions about each abstract observation $\bar{o} \in \bar{O}$:

$$P(\bar{o} \mid h, a) \quad = \quad P(\bar{o} \mid \chi(h), g(a))$$

Lemma 2.1. *If a POMDP homomorphism satisfying Equations 2.10, 2.11 and 2.12 exists for the observation mapping κ, $\bar{O}$ is self-sufficient.*

Proof.

$$
\begin{aligned}
\forall h \in H_M, P(\bar{o} \mid h, a) &= \sum_{s \in S} b_h(s) \cdot \sum_{o \in \bar{o}} P(o \mid s, a) \\
&= \sum_{s \in S} b_h(s) \cdot P(\bar{o} \mid s, a) && \text{Definition of } P(\bar{o} \mid s, a) \\
&= \sum_{s \in S} P(\bar{o} \mid f(s), g(a)) \cdot b_h(s) && \text{Equation 2.10} \\
&= \sum_{\bar{s} \in \bar{S}} \sum_{s \in \bar{s}} P(\bar{o} \mid f(s), g(a)) \cdot b_h(s) && f \text{ partitions } S \\
&= \sum_{\bar{s} \in \bar{S}} P(\bar{o} \mid \bar{s}, g(a)) \cdot \sum_{s \in \bar{s}} b_h(s) && \text{Definition of } f \\
&= \sum_{\bar{s} \in \bar{S}} P(\bar{o} \mid \bar{s}, g(a)) \cdot \bar{b}_{\chi(h)}(\bar{s}) && \text{Equation 2.12} \\
&= P(\bar{o} \mid \chi(h), g(a))
\end{aligned}
$$

□

As previously mentioned, any POMDP can be transformed into a "history" CMP (though the state set of the CMP is possibly infinite in size). The CMP uses the set of reachable histories (H_M) as the state set: $C_H = (H_M, A, T_H)$. The transition function T_H for two histories h and hao is defined as follows:

$$T_H(h, a, hao) = P(o|h, a)$$

in all other cases $T_H(h, a, h') = 0$.

Lemma 2.2. *If f, g and κ form a POMDP homomorphism satisfying Equations 2.9 -2.12, then χ and g form a CMP homomorphism over the history CMP C_M, with output set $Y = Z$ and output function $\Upsilon(h, a, z) = \sum_{o \in O} P(o \mid h, a) \cdot \zeta(o, z)$.*

Proof. Output distribution:

$$
\begin{aligned}
\forall h \in H_M, \Upsilon(h,a,z) &= \sum_{o \in O} P(o \mid h,a) \cdot \zeta(o,z) \\
&= \sum_{\bar{o} \in \bar{O}} \bar{\zeta}(\kappa(o),z) \cdot \sum_{o \in \bar{o}} P(o \mid h,a) && \text{Equation 2.9} \\
&= \sum_{\bar{o} \in \bar{O}} \bar{\zeta}(\kappa(o),z) \cdot P(\kappa(o) \mid h,a) \\
&= \sum_{\bar{o} \in \bar{O}} \bar{\zeta}(\kappa(o),z) \cdot P(\bar{o} \mid \chi(h), g(a)) && \text{Lemma 2.1} \\
&= \bar{\Upsilon}(\chi(h), g(a), z)
\end{aligned}
$$

Transitions:

$$
\begin{aligned}
\bar{T}_H(\chi(h), g(a), \chi(hao)) &= P(\kappa(o) \mid \chi(h), g(a)) \\
&= P(\kappa(o) \mid h,a) \quad \text{Lemma 2.1} \\
&= \sum_{o' \in [o]_\kappa} P(o' \mid h,a) \\
&= \sum_{hao' \in [hao]_\chi} T_H(h,a,hao')
\end{aligned}
$$

□

A similar CMP $C_B = \{B_M, A, T_B\}$ can be constructed using belief states as the state set. The same result can be proven for this belief state CMP, using much the same proof.

One of the most important properties of a homomorphism is that it preserves the optimal value function, and thus the optimal policy.

Lemma 2.3. *A POMDP homomorphism preserves the optimal value function for any reward function $r : Z \to \mathbb{R}$ which is a function of the POMDP output set.*

Proof. Since χ, g is a homomorphism for the CMP C_H:

$$\forall h \in H_M, V(h) = \bar{V}(\chi(h)) \qquad \text{Lemma 2.2}$$

Where V is the value function of r in the history CMP C_H, and $\bar{V}$ is the value function of r in the abstract history CMP $\bar{C}_{\bar{H}}$ which results from the application of the history state mapping χ and action mapping g.

In any POMDP, the value of a belief state is the value of the history corresponding to that belief state:

$$\forall h \in H_M, V(h) = V(b_h). \qquad \text{POMDP definition}$$

This is true in the abstract POMDP as well:

$$\forall h \in H_M, \bar{V}(\chi(h)) = \bar{V}(b_{\chi(h)}). \qquad \text{POMDP definition}$$

Putting these facts together:

$$\begin{aligned} V(b_h) &= V(h) \\ &= \bar{V}(\chi(h)) \\ &= \bar{V}(\bar{b}_{\chi(h)}) \end{aligned}$$

And thus the value function for M can be lifted from $\bar{M}$. □

The general outline of the entire POMDP Homomorphism finding algorithm is shown in Procedure 2.3.1. The next several sections will build up this algorithm in stages, starting with the problem of evaluating a given observation mapping κ, and presenting two reasonable alternatives to Equation 2.12. The problems addressed, in order of appearance are:

Procedure 2.3.1 Find POMDP Homomorphim($M = (S, A, T, O, \Omega)$, $output = (Z, \zeta)$)

calculate $\bar{O}$ to predict Z
calculate f and g to support $\bar{O}$
while κ, f, g is not a homomorphism **do**
  improve/evaluate κ
  calculate f and g to support $\bar{O}$
return κ, f, g

- Evaluate κ when for all observation/state/action combinations, $P(o \mid s, a) > 0$ (Shadow Model test).
- Evaluate κ when there are some observation probabilities which are 0 (Shadow Compatibility test).
- Improve κ based on its evaluation (using either algorithm)

Both of the algorithms for evaluating a given κ construct the state and action maps f and g in the process of evaluation. Both algorithms also construct data structures which will be used to refine κ. Both of these evaluation algorithms run in polynomial time in the worst case, though the assumption that $P(o \mid s, a) > 0$ allows the first algorithm to be somewhat faster in the worst case than the second.

Figure 2.1 illustrates accept sets for these two algorithms. The Shadow Model test accepts a smaller set of abstractions, particularly where the requirement that $P(o \mid s, a) > 0$ is not met. In fact, in many cases where this requirement is not met, this test rejects all possible abstractions — up to and including the abstraction formed by the identity mapping. The Shadow Compatibility test has a larger accept set, indicating that in some cases it may accept smaller abstract models. This test is also more complete in that it will accept at least one mapping function for any given POMDP, since the identity mapping is always accepted.

The problem of finding an optimal refinement of κ under this framework will be shown to be NP-hard in the worst case, when $P(o \mid s, a) > 0$ is not satisfied for all state, actions and observations. However, the data structures used in evaluating κ can be used to narrow

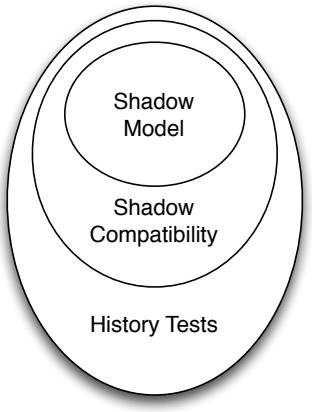


Figure 2.1. Diagram illustrating the overlap between the accept sets for the two approaches outlined in this chapter. The Compatibility Model approach is more likely to find a smaller abstract model than the Shadow Model since its accept set is a superset of the Shadow Model accept set.

the search for better observation maps by pinpointing aspects of the abstract model that should be improved. As with the observation mapping function refinement algorithm in Talvitie et al. (2008) (discussed in Section 1.1.2), the κ refinement algorithm identifies a set of acceptable observation mapping functions. Using the Shadow Model and Shadow Compatibility test data structures, this set can be found in polynomial time (rather than exponential). However, the problem of choosing the specific observation mapping function in this set for which the smallest abstract state and action sets would be required is still NP-complete in the worst case. Any heuristic that chooses an element from the set of acceptable observation mapping functions will produce an accurate model, but it may not be the smallest model satisfying the acceptance criteria. Nevertheless, identifying this set in polynomial time is an improvement over the existing literature.

2.4 Evaluating an Observation Map

Evaluating a given observation map to determine whether it corresponds to a valid POMDP homomorphism is central to the task of building an abstract observation function.

In this section it will be assumed that every observation occurs in every state with some probability. That is, for all states s, actions a and observations o:

$$P(o \mid s, a) > 0. \tag{2.13}$$

Section 2.5 will relax this assumption.

The algorithm for evaluating the observation map κ is shown in Procedure 2.4.1. If Procedure 2.4.1 succeeds, f, g and κ form a valid homomorphic reduction of M. This test is practical: it does not require the examination of every history in H_M, as a naive examination of constraint Equation 2.12 might. However, if the test fails, it is still possible that Equations 2.12 could be satisfied, and that examining every history in H_M would have verified this fact.

At a high level, the algorithm has 3 parts:

- construct the abstract model
- construct a *shadow* model
- determine whether the shadow and abstract models are independent.

The full observation set consists of two types of information: information that is used by the abstract model, and information that is not. The shadow model keeps track of all of the information that is ignored by the abstract model (see Figure 2.5 for an illustration of the relationship). The belief states of the shadow model form a sufficient statistic for the observation information that is ignored by the abstract model. If the shadow belief state cannot be used to improve abstract belief state predictions, then the unused observation information also cannot be used to improve abstract state predictions.

The shadow model (Figure 2.2(c)) treats the abstract observation as a node with no parents, similar to an action, and observes the true observation o according to the conditional probability $P(o \mid s, a, \bar{o})$.

Procedure 2.4.1 Evaluate observation mapping($M = (S, A, T, O, \Omega), \bar{O}, \kappa : O \to \bar{O}$)

verify $\zeta(o, z) = \bar{\zeta}(\kappa(o), z)$ directly

// Construct an abstract CMP model for κ
$C \leftarrow (S, A, T)$
$\Omega_\chi : S \times A \times \bar{O} \to [0, 1]$
$\Omega_\chi(s, a, \bar{o}) \stackrel{\text{def}}{=} P(\bar{o} \mid s, a)$
$f, g \leftarrow$ findCMPHomomorphism($C, output = (\bar{O}, \Omega_\chi)$)

// Construct an abstract "shadow" CMP for κ
$C \leftarrow (S, A, T)$
$\Omega_\xi : S \times A \times O \to [0, 1]$
$\Omega_\xi(s, a, o) \stackrel{\text{def}}{=} \frac{P(o|s,a)}{P(\kappa(o)|s,a)}$
$f_\xi, g_\xi \leftarrow$ findCMPHomomorphism($C, output = (O, \Omega_\xi)$)

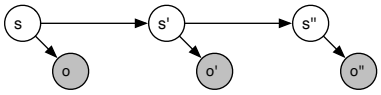
// determine whether the two CMP models operate independently of one another
for all $s \in S, a \in A, s' \in S$ **do**
  $P(f(s') \wedge f_\xi(s')|s, a) \Leftarrow \sum_{s'' \in [s']_f \cap [s']_{f_\xi}} P(s'' \mid s, a)$
  if $\neg\left(P(f(s') \wedge f_\xi(s')|s, a) = P(f(s')|f(s), g(a)) \cdot P(f_\xi(s')|f_\xi(s), g_\xi(a))\right)$ **then**
    return false
for all $s \in S$ **do**
  $b_\lambda(f(s) \wedge f_\xi(s)) \Leftarrow \sum_{s'' \in [s]_f \cap [s]_{f_\xi}} b_\lambda(s'')$
  if $\neg\left(b_\lambda(f(s) \wedge f_\xi(s)) = b_\lambda(f(s)) \cdot b_\lambda(f_\xi(s))\right)$ **then**
    return false

// Test passes
return f, g, κ

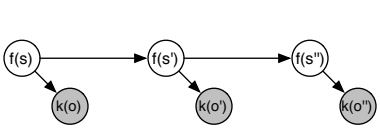
If the abstract and shadow states are independent over time, then the true observation o does not add any information that would help predict the abstract observations.

2.4.1 Abstract Model

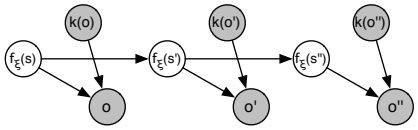
The second step of Procedure 2.4.1 constructs a candidate abstract model for M. Recall that Equations 2.10 and 2.11 resemble Equations 1.5 and 1.6, the constraints of a CMP Homomorphism. Procedure 2.4.1 constructs an abstract POMDP via a CMP Homomorphism, with one caveat: no state specific action mapping functions. Rather than a set of functions $g_s : A \to \bar{A}$ for each state s, there must be a single, global action mapping func-



(a) POMDP Model



(b) Abstract Model



(c) Abstract Shadow Model

Figure 2.2. The abstract model and the abstract shadow model (see Section 2.4.4), illustrated as Baysian Networks. Three "rolled out" time steps are shown. Action nodes are not shown. Shaded nodes are observed variables.

tion $g : A \to \bar{A}$. State-specific action maps cannot be used in this context because they could introduce conflicts in the policy when the state is uncertain. Procedure 2.4.2 outlines the algorithm for finding a CMP Homomorphism, when state-specific action maps are not used. Define:

$$O_\chi = \bar{O}$$
$$\Omega_\chi(s, a, \bar{o}) = \sum_{o \in \bar{o}} \Omega(s, a, o).$$

The state and action mappings f and g constructed in Procedure 2.4.1 form a CMP homomorphism for the CMP (S, A, T) with output $(\bar{O}, \Omega_\chi)$. This CMP Homomorphism preserves the following constraints in the abstract model:

$$P(\bar{o} \mid f(s), g(a)) = P(\bar{o} \mid s, a) \qquad \text{Equation 2.10 (from Equation 1.5)}$$
$$P(f(s_j) \mid f(s_i), g(a)) = \sum_{s \in [s_j]_f} P(s \mid s_i, a) \qquad \text{Equation 2.11 (from Equation 1.6)}$$

Let M_χ be a POMDP with abstract observations, but the original state and action sets:

Procedure 2.4.2 findCMPHomomorphism($C = (S, A, T)$, $output = (Y, \Upsilon)$)

// Partition S to create f_0 and $\bar{S}_0$ such that
$f_0(s_i) = f_0(s_j) \iff P(y \mid s_i, a) = P(y \mid s_j, a) \quad (\forall a \in A, \forall y \in Y)$
$\tau = 0$
repeat
  // Partition A to create $g_{\tau+1}$ and $\bar{A}_{\tau+1}$ such that
  $g_{\tau+1}(a_i) = g_{\tau+1}(a_j) \iff P(\bar{s} \mid s, a_i) = P(\bar{s} \mid s, a_j) \quad (\forall s \in S, \bar{s} \in \bar{S}_\tau)$
  // Partition S to create $f_{\tau+1}$ and $\bar{S}_{\tau+1}$ such that
  $f_{\tau+1}(s_i) = f_{\tau+1}(s_j) \iff P(\bar{s} \mid s_i, \bar{a}) = P(\bar{s} \mid s_j, \bar{a}) \quad (\forall \bar{a} \in \bar{A}_{\tau+1}, \bar{s} \in \bar{S}_\tau)$
  // and
  $f_{\tau+1}(s_i) = f_{\tau+1}(s_j) \iff P(y \mid s_i, a) = P(y \mid s_j, a) \quad (\forall a \in A, \forall y \in Y)$
  $\tau \leftarrow \tau + 1$
until $\bar{S}_\tau = \bar{S}_{\tau-1}$

return f, g

$$M_\chi = (S, A, T, O_\chi, \Omega_\chi). \tag{2.14}$$

The belief state for abstract history $\chi(h)$ (defined in Equation 2.8) in M_χ will be denoted $b_{\chi(h)}$, and the initial belief state is $b_{\chi(\lambda)} = b_\lambda$.

If $f : S \to \bar{S}$ and $g : A \to \bar{A}$, let $\bar{M}_\chi$ denote the candidate abstract POMDP:

$$\bar{M}_\chi = (\bar{S}, \bar{A}, \bar{T}, O_\chi, \bar{\Omega}_\chi). \tag{2.15}$$

where $\bar{T}$ is consistent with Equation 2.11 and $\bar{\Omega}_\chi$ is consistent with Equation 2.10. The belief state for abstract history $\chi(h)$ in $\bar{M}_\chi$ will be denoted $\bar{b}_{\chi(h)}$. The initial belief state $\bar{b}_{\chi(\lambda)}$ is defined:

$$\bar{b}_{\chi(\lambda)}(\bar{s}) = \sum_{s \in \bar{s}} b_\lambda(s). \tag{2.16}$$

At this point in the algorithm, it is not yet possible to determine whether the candidate abstract model $\bar{M}_\chi$ is a homomorphic abstraction for M. However, $\bar{M}_\chi$ can be shown to be a homomorphic abstraction for M_χ. The Lemma 2.4 shows that the mapping from M_χ to $\bar{M}_\chi$ satisfies Equation 2.12, in addition to Equations 2.10 and 2.11.

Lemma 2.4. *For any $h \in H_M$ and $\bar{s} \in \bar{S}$:*

$$\sum_{s \in \bar{s}} b_{\chi(h)}(s) = \bar{b}_{\chi(h)}(\bar{s})$$

Proof. By Structural Induction on H_M.

Base case (λ): By definition, for any $\bar{s} \in \bar{S}$:

$$\bar{b}_{\chi(\lambda)}(\bar{s}) = \sum_{s \in \bar{s}} b_\lambda(s) = \sum_{s \in \bar{s}} b_{\chi(\lambda)}(s).$$

Inductive step (h to hao):

$$\begin{aligned}
\sum_{s' \in \bar{s}'} b_{\chi(hao)}(s) &= \frac{\sum_{s' \in \bar{s}'} P(\kappa(o)|s', a) \cdot \sum_{s \in S} P(s'|s, a) \cdot b_{\chi(h)}(s)}{\sum_{s' \in S} P(\kappa(o)|s', a) \cdot \sum_{s \in S} P(s'|s, a) \cdot b_{\chi(h)}(s)} \\
&= \frac{P(\kappa(o)|\bar{s}', g(a)) \cdot \sum_{s \in S} P(\bar{s}'|s, a) \cdot b_{\chi(h)}(s)}{\sum_{\bar{s}' \in \bar{S}} P(\kappa(o)|\bar{s}', g(a)) \cdot \sum_{s \in S} P(\bar{s}'|s, a) \cdot b_{\chi(h)}(s)} && \text{Eqn 2.10} \\
&= \frac{P(\kappa(o)|\bar{s}', g(a)) \cdot \sum_{\bar{s} \in \bar{S}} P(\bar{s}'|\bar{s}, g(a)) \cdot \sum_{s \in \bar{s}} b_{\chi(h)}(s)}{\sum_{\bar{s}' \in \bar{S}} P(\kappa(o)|\bar{s}', g(a)) \cdot \sum_{\bar{s} \in \bar{S}} P(\bar{s}'|\bar{s}, g(a)) \cdot \sum_{s \in \bar{s}} b_{\chi(h)}(s)} && \text{Eqn 2.11} \\
&= \frac{P(\kappa(o)|\bar{s}', g(a)) \cdot \sum_{\bar{s} \in \bar{S}} P(\bar{s}'|\bar{s}, g(a)) \cdot \bar{b}_{\chi(h)}(\bar{s})}{\sum_{\bar{s}' \in \bar{S}} P(\kappa(o)|\bar{s}', g(a)) \cdot \sum_{\bar{s} \in \bar{S}} P(\bar{s}'|\bar{s}, g(a)) \cdot \bar{b}_{\chi(h)}(\bar{s})} && \text{Ind. hyp.} \\
&= \bar{b}_{\chi(hao)}(\bar{s}')
\end{aligned}$$

□

This Lemma shows that the abstract POMDP maintains its abstract belief state as accurately as the abstract observations allow. Now $\bar{M}_\chi$ must be tested to determine whether its state estimates are as accurate as they would be if the full observation set were accessible. This requires the construction of the *shadow* model (Figure 2.2(c)).

2.4.2 Abstract and Shadow Models: Two Examples

The shadow model is somewhat easier to understand when the observations are factored into features. Consider the gridworld shown in Figure 2.3(a). The POMDP for this domain is defined as follows:

States: Each square in Figure 2.3(a) represents a location. State is the agent's location.

Actions: up, $down$, $left$, $right$

Transitions: Actions fail with a small probability ϵ_a. Failure results in no change to the state.

Observations: Factored, with two features:

- boolean $cheese$ or $\neg cheese$ feature
- $color$ feature: $lightgrey$, $grey$, $black$

Observation Function: In each state, the agent observes only the features of the square it currently occupies. Each feature takes on a random noise value with some small probability ϵ_o.

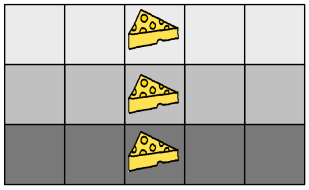
Initial Belief State: Uniform probability of being in each state.

Take ζ to be the boolean $cheese$ indicator feature. Other observation features may be relevant to predicting ζ or not relevant. Consider the following observation map, directly determined from ζ:

- Abstract observation 1: all observations with $cheese$
- Abstract observation 2: all observations with $\neg cheese$.

Figure 2.3(b) is a candidate abstract POMDP model for the this observation function, constructed via a CMP Homomorphism for (S, A, T), with output function $cheese/\neg cheese$. Each abstract state represents a cluster of three states, grouped according to column. The row coordinate of the state is not helpful in predicting the output function, and is ignored in the abstract model.

Figure 2.3(c), on the other hand, is the "shadow" model left behind by the abstract model for $cheese$. Without the $cheese$ feature the only remaining feature is color, so that the shadow observations are:



(a) Domain

(b) Abstract Model: Cheese Feature

(c) AbstractModel: Color Feature

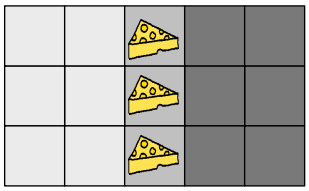
Figure 2.3. In this gridworld (2.3(a)), there are two features. The cheese locations are predicted by the column of the state, and the color of the location is predicted by the row of the state. Abstract models for each feature are shown in 2.3(b) and 2.3(c).

- Shadow observation 1: all observations with $lightgrey$
- Shadow observation 2: all observations with $grey$
- Shadow observation 3: all observations with $black$.

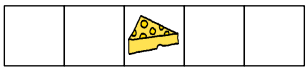
The abstract shadow model is shown in Figure 2.3(c), and is formed by a CMP Homomorphism for the CMP (S, A, T), with output function $color$. In this case, the abstract model only retains row information in the abstract state and ignores column information.

Since Figure 2.3(b) (the abstract model for $cheese$) uses column information and Figure 2.3(c) (the shadow model for $cheese$) uses row information, and the row and column state features do not affect one another in this gridworld, the abstract and shadow models should be verifiably independent in the final test.

In Figure 2.4, on the other hand, the color of the locations varies with column, not row. This implies that color information could be used to improve estimated column location,



(a) Domain



(b) Abtract Model: Cheese Feature

(c) Abstract Model: Color Feature

Figure 2.4. In this gridworld (2.4(a)), there are two features. The cheese locations are predicted by the column of the state. The color of the state in this case is also predicted by the column of the state. Abstract models for each feature are shown in 2.4(b) and 2.4(c).

and thus predictions about cheese. A test of the $cheese/\neg cheese$ observation abstraction should reveal the fact that something useful has been left out of the abstract observation function.

The abstract model for the $cheese/\neg cheese$ observation abstraction for this domain is shown in Figure 2.4(b), and the shadow model for this abstraction is shown in Figure 2.4(c). In this case the states of the abstract and shadow models are perfectly correlated (i.e. not independent).

2.4.3 Shadow Model

In the examples of Figures 2.3 and 2.4, observations are defined by observation features, and in addition these features are independent of each other given the state and action. Under these conditions, the shadow observation set can be defined via the set of unused observation features. However, in general this may not be the case.

Consider some arbitrary belief state b. Each update $P(s' \mid b, a, o)$ for the next state s', action a and observation o can be calculated in stages. The POMDP update rule has

two stages: first $P(s' \mid b, a)$ is calculated (Equation 2.2), then $P(s' \mid b, a, o)$ (Equation 2.3). To construct the shadow observation function, notice that the update can be further deconstructed into three stages using κ:

1. calculate: $P(s'|b, a)$ for all $s' \in S$
2. use the results of 1 to calculate: $P(s'|b, a, \kappa(o))$
3. use the results of 2 to calculate: $P(s'|b, a, o)$

Separating the POMDP update rule into these 3 component parts yields:

$$P(s' \mid b, a, o) = \overbrace{P(s' \mid b, a)}^{a \text{ update}} \cdot \frac{\overbrace{P(\kappa(o) \mid s', a)}^{\kappa(o) \text{ update}}}{P(\kappa(o) \mid b, a)} \cdot \overbrace{\frac{P(o \mid s', a)}{P(o \mid b, a)} \cdot \frac{P(\kappa(o) \mid b, a)}{P(\kappa(o) \mid s', a)}}^{o \text{ update}}.$$

The model M_χ uses updates 1 and 2, but not update 3:

$$\begin{aligned} P(s \mid b, a, \kappa(o)) &= \overbrace{P(s' \mid b, a)}^{a \text{ update}} \cdot \frac{\overbrace{P(\kappa(o) \mid s', a)}^{\kappa(o) \text{ update}}}{P(\kappa(o) \mid b, a)} \\ &= \frac{P(\kappa(o) \mid s', a) \sum_s P(s' \mid s, a) \cdot b(s)}{\sum_{s'} P(\kappa(o) \mid s', a) \sum_s P(s' \mid s, a) \cdot b(s)} \end{aligned}$$

which is just the POMDP update rule (Equation 2.1) for a POMDP that has abstract observations (M_χ, defined in Equation 2.14).

The shadow model, on the other hand, is defined by updates 1 and 3, but ignores update 2. The shadow model update rule treats the abstract observation $\kappa(o)$ as an action, rather than an observation to be predicted (see Figure 2.2(c)). The action set is therefore $A \times \bar{O}$, and individual "actions" can be written $\langle a, \bar{o} \rangle$, where a is an action and $\bar{o}$ an abstract observation. The shadow model update rule is:

$$P(s' \mid b, \langle a, \kappa(o) \rangle, o) \propto \overbrace{P(s' \mid b, a)}^{a \text{ update}} \cdot \overbrace{\frac{P(o \mid s', a)}{P(o \mid b, a)} \cdot \frac{P(\kappa(o) \mid b, a)}{P(\kappa(o) \mid s', a)}}^{o \text{ update}}$$

Since the term $\frac{P(\kappa(o)|b,a)}{P(o|b,a)}$ does not vary with s', this simplifies to:

$$P(s' \mid b, \langle a, \kappa(o)\rangle, o) \propto \overbrace{P(s' \mid b, a)}^{a \text{ update}} \cdot \overbrace{\frac{P(o \mid s', a)}{P(\kappa(o) \mid s', a)}}^{o \text{ update}}$$

Normalizing to get a probability distribution over states s':

$$\begin{aligned} P(s' \mid b, \langle a, \kappa(o)\rangle, o) &= \frac{P(s' \mid b, a) \cdot \frac{P(o|s',a)}{P(\kappa(o)|s',a)}}{\sum_{s'} P(s' \mid b, a) \cdot \frac{P(o|s',a)}{P(\kappa(o)|s',a)}} \\ &= \frac{\frac{P(o|s',a)}{P(\kappa(o)|s',a)} \sum_s P(s' \mid s, a) \cdot b(s)}{\sum_{s'} \frac{P(o|s',a)}{P(\kappa(o)|s',a)} \sum_s P(s' \mid s, a) \cdot b(s)} \end{aligned}$$

This is similar to the POMDP update rule (Equation 2.1), with observation probability function $\frac{P(o|s',a)}{P(\kappa(o)|s',a)}$. The shadow model is a POMDP with this observation probability function.

In the special case represented by Figures 2.3 and 2.4 the observation set is factored into independent features. Under these conditions, the observation ratio $\frac{P(o|s',a)}{P(\kappa(o)|s',a)}$ reduces to the probability of the unused feature set. Take, for example, the two observations $o_1 = lightgrey \wedge cheese$ and $o_2 = lightgrey \wedge \neg cheese$. When the abstract observation includes only $cheese/\neg cheese$ information, the observation ratios for boh o_1 and o_2 reduce to the probability of the $lightgrey$ feature. For any state s and action a:

$$\begin{aligned} \frac{P(o_1 \mid s, a)}{P(\kappa(o_1) \mid s, a)} &= \frac{P(lightgrey, cheese \mid s, a)}{\sum_{c \in colors} P(c, cheese \mid s, a)} \\ &= \frac{P(lightgrey \mid s, a) \cdot P(cheese \mid s, a)}{P(cheese \mid s, a)} \\ &= P(lightgrey \mid s, a) \end{aligned}$$

and

$$\begin{aligned}\frac{P(o_2 \mid s,a)}{P(\kappa(o_2) \mid s,a)} &= \frac{P(lightgrey, \neg cheese \mid s,a)}{\sum_{c \in colors} P(c, \neg cheese \mid s,a)} \\ &= \frac{P(lightgrey \mid s,a) \cdot P(\neg cheese \mid s,a)}{P(\neg cheese \mid s,a)} \\ &= P(lightgrey \mid s,a).\end{aligned}$$

where $colors = \{lightgrey, grey, black\}$.

However, if appropriate observation features are not provided, or if the observation features are not conditionally independent given s, a, the ratio may not simplify in this manner and the ratio $\frac{P(o|s',a)}{P(\kappa(o)|s',a)}$ must be used directly.

2.4.4 Abstract Shadow Model

The abstract shadow model should be independent of the abstract model if possible, while preserving the observation probability ratio accurately. In this chapter, the abstract shadow model is constructed via a CMP Homomorphism (later sections will explore other definitions). Define the shadow observation set and observation function:

$$\begin{aligned}O_\xi &= O \\ \Omega_\xi(s, \langle a, \kappa(o)\rangle, o) &= \frac{P(o \mid s,a)}{P(\kappa(o) \mid s,a)}\end{aligned}$$

Recall that in this section it is assumed that all observations occur with some non-zero probability in every state. This implies that $\frac{P(o|s,a)}{P(\kappa(o)|s,a)}$ is always well defined.

The state and action mappings $f_\xi : S \to \tilde{S}$ and $g_\xi : A \to \tilde{A}$ constructed in Procedure 2.4.1 form a CMP homomorphism for the CMP (S, A, T) with output (O_ξ, Ω_ξ). This CMP Homomorphism preserves the following constraints in the abstract model:

$$P(o \mid f_\xi(s), g_\xi(a)) = \frac{P(o \mid s,a)}{P(\kappa(o) \mid s,a)} \tag{2.17}$$

$$P(f_\xi(s_j) \mid f_\xi(s_i), g_\xi(a)) = P(f_\xi(s_j) \mid s_i, a) \tag{2.18}$$

Define the shadow POMDP M_ξ as:

$$M_\xi = (S, A_\xi, T_\xi, O_\xi, \Omega_\xi) \tag{2.19}$$

where

$$A_\xi = A \times \bar{O}$$
$$T_\xi(s, \langle a, \bar{o} \rangle, s') = T(s, a, s').$$

Let the shadow history mapping function $\xi : H_M \to H_{M_\xi}$ be defined as:

$$\xi(\lambda) = \lambda \tag{2.20}$$
$$\xi(hao) = \xi(h)\langle a\kappa(o) \rangle o.$$

The belief states $b_{\xi(h)}$ for the shadow POMDP are defined according to the standard belief state definition, with initial belief b_λ.

The state and action mapping functions f_ξ and g_ξ can be used to build an abstract shadow model:

$$\tilde{M}_\xi = (\tilde{S}, \tilde{A}, \tilde{T}, O, \tilde{O}). \tag{2.21}$$

where $\tilde{T}$ is consistent with Equation 2.18 and $\tilde{O}$ is consistent with Equation 2.17. The abstract shadow belief state $\tilde{b}_{\xi(h)}$ is maintained according to the standard POMDP update rules on this POMDP, with initial belief state definition:

$$\tilde{b}_{\xi(\lambda)}(\tilde{s}) = \sum_{s \in \tilde{s}} b_\lambda(s). \tag{2.22}$$

for any $\tilde{s}$ in $\tilde{S}$.

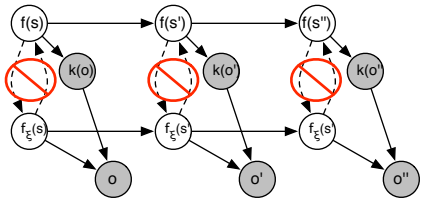


Figure 2.5. Abstract and shadow model interactions, shown as a Bayesian Network. Three time steps are shown. Shaded nodes are observed. Action nodes are not shown. The states of the two models are independent if the observations o, o', o'', etc, can be accurately predicted without dependency edges between the state nodes of the two models.

2.4.5 Independence of Shadow and Abstract Models

The final stage of Procedure 2.4.1 verifies that the states of the abstract model $\bar{M}_\chi$ and abstract shadow model $\tilde{M}_\xi$ are independent, as shown in Figure 2.5. There are two steps to this test: checking the next state distribution for each state and action pair, and checking the initial belief distribution. If the following equation is satisfied for all pairs $\bar{s} \in \bar{S}$ and $\tilde{s} \in \tilde{S}$:

$$\sum_{s \in \bar{s} \cap \tilde{s}} b_\lambda(s) = \bar{b}_\lambda(\bar{s}) \cdot \tilde{b}_\lambda(\tilde{s}) \tag{2.23}$$

where $s \in \bar{s} \cap \tilde{s}$ includes all states s for which $f(s) = \bar{s}$ and $f_\xi(s) = \tilde{s}$, then the initial belief state passes the independence test.

If following equation must be satisfied for every state s, action a and next state s':

$$\sum_{s' \in \bar{s}' \cap \tilde{s}'} P(s' \mid s, a) = P(\bar{s}' \mid f(s), g(a)) \cdot P(\tilde{s}' \mid f_\xi(s), g_\xi(a)) \tag{2.24}$$

then the transition function passes the test. The final steps of Procedure 2.4.1 implement these two tests.

These two properties (Equations 2.23 and 2.24) imply that the belief state factors into abstract and shadow components after every history h in H_M.

Theorem 2.5. *Equations 2.23 and 2.24 imply that for any history h in H_M, $\sum_{s\in\bar{s}\cap\tilde{s}} b_h(s) = \bar{b}_{\chi(h)}(\bar{s}') \cdot \tilde{b}_{\xi(h)}(\tilde{s}')$.*

Proof. Proof by Structural Induction on H_M.

Basis step (λ): By Equation 2.23.

The Inductive step is in two parts. First, the step from b_h to b_{ha} then, the step from b_{ha} to b_{hao}.

Inductive Step (h to ha): Assume that $\sum_{s\in\bar{s}\cap\tilde{s}} b_h(s) = \bar{b}_{\chi(h)}(\bar{s}') \cdot \tilde{b}_{\xi(h)}(\tilde{s}')$. For any $\bar{s}' \in \bar{S}$ and $\tilde{s}' \in \tilde{S}$,

$$
\begin{aligned}
\sum_{s'\in\bar{s}'\cap\tilde{s}'} b_{ha}(s') &= \sum_{s'\in\bar{s}'\cap\tilde{s}'} \sum_{s\in S} P(s'|s,a) b_h(s) && \text{Equation 2.2} \\
&= \sum_{s\in S} b_h(s) \cdot \sum_{s'\in\bar{s}'\cap\tilde{s}'} P(s'|s,a) \\
&= \sum_{s\in S} b_h(s) \cdot P(\bar{s}'|f(s),g(a)) \cdot P(\tilde{s}'|f_\xi(s),g_\xi(a)) && \text{Equation 2.24} \\
&= \sum_{\bar{s}\in\bar{S}} \sum_{\tilde{s}\in\tilde{S}} P(\bar{s}'|\bar{s},g(a)) \cdot P(\tilde{s}'|\tilde{s},g_\xi(a)) \cdot \sum_{s\in\bar{s}\cap\tilde{s}} b_h(s) && f \text{ and } f_\xi \text{ partition } S \\
&= \sum_{\bar{s}\in\bar{S}} P(\bar{s}'|\bar{s},g(a)) \bar{b}_{\chi(h)}(\bar{s}) \cdot \sum_{\tilde{s}\in\tilde{S}} P(\tilde{s}'|\tilde{s},g_\xi(a)) \cdot \tilde{b}_h(\tilde{s}) && \text{Assumption} \\
&= \bar{b}_{\chi(ha)}(\bar{s}') \cdot \tilde{b}_{\xi(ha)}(\tilde{s}') && \text{Equation 2.2}
\end{aligned}
$$

Inductive Step (ha to hao): For any $\bar{s}' \in \bar{S}$ and $\tilde{s}' \in \tilde{S}$,

$$
\begin{aligned}
\sum_{s' \in \bar{s}' \cap \tilde{s}'} b_{hao}(s') &= \sum_{s' \in \bar{s}' \cap \tilde{s}'} \frac{P(o \mid s', a) \cdot b_{ha}(s')}{\sum_{s' \in S} P(o \mid s', a) \cdot b_{ha}(s')} \\
&= \frac{\sum_{s' \in \bar{s}' \cap \tilde{s}'} P(o \mid s', a) \cdot b_{ha}(s')}{\sum_{\bar{s}' \in \bar{S}} \sum_{\tilde{s}' \in \tilde{S}} \sum_{s' \in \bar{s}' \cap \tilde{s}'} P(o \mid s', a) \cdot b_{ha}(s')} \\
&= \frac{P(\kappa(o) \mid \bar{s}', g(a)) \cdot P(o \mid \tilde{s}', g_\xi(a)) \cdot \sum_{s' \in \bar{s}' \cap \tilde{s}'} b_{ha}(s')}{\sum_{\bar{s}' \in \bar{S}} \sum_{\tilde{s}' \in \tilde{S}} P(\kappa(o) | \bar{s}', g(a)) \cdot P(o | \tilde{s}', g_\xi(a)) \cdot \sum_{s' \in \bar{s}' \cap \tilde{s}'} b_{ha}(s')} \\
&= \frac{P(\kappa(o) \mid \bar{s}', g(a)) \cdot P(o \mid \tilde{s}', g_\xi(a)) \cdot \bar{b}_{\chi(h)}(\bar{s}') \cdot \tilde{b}_{\xi(h)}(\tilde{s}')}{\sum_{\bar{s}' \in \bar{S}} \sum_{\tilde{s}' \in \tilde{S}} P(\kappa(o) | \bar{s}', g(a)) \cdot P(o | \tilde{s}', g_\xi(a)) \cdot \bar{b}_{\chi(h)}(\bar{s}') \cdot \tilde{b}_{\xi(h)}(\tilde{s}')} \\
&= \frac{P(\kappa(o) \mid \bar{s}', g(a)) \cdot \bar{b}_{\chi(h)}(\bar{s}')}{\sum_{\bar{s}' \in \bar{S}} P(\kappa(o) | \bar{s}', g(a)) \cdot \bar{b}_{\chi(h)}(\bar{s}')} \cdot \frac{P(o \mid \tilde{s}', g_\xi(a)) \cdot \tilde{b}_{\xi(h)}(\tilde{s}')}{\sum_{\tilde{s}' \in \tilde{S}} P(o | \tilde{s}', g_\xi(a)) \cdot \tilde{b}_{\xi(h)}(\tilde{s}')} \\
&= \bar{b}_{\chi(hao)}(\bar{s}') \cdot \tilde{b}_{\xi(hao)}(\tilde{s}')
\end{aligned}
$$

□

With this theorem in hand, it is straightforward to prove that Equation 2.12 is satisfied if Equations 2.23 and 2.24 are satisfied.

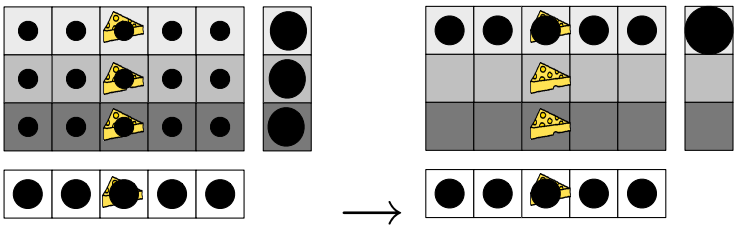
Theorem 2.6. *If* $\forall h \in H_M, \sum_{s \in \bar{s} \cap \tilde{s}} b_h(s) = \bar{b}_{\chi(h)}(\bar{s}') \cdot \tilde{b}_{\xi(h)}(\tilde{s}')$*, then Equation 2.12 is true (for all histories* h*,* $\bar{b}_{\chi(h)}(f(s)) = \sum_{s' \in [s]_f} b_h(s')$*).*

Proof.

$$
\begin{aligned}
\sum_{s' \in \bar{s}'} b_h(s') &= \sum_{\tilde{s}' \in \tilde{S}} \sum_{s' \in \bar{s}' \cap \tilde{s}'} b_h(s') \\
&= \sum_{\tilde{s}' \in \tilde{S}} \bar{b}_{\chi(h)}(\bar{s}') \cdot \tilde{b}_{\xi(h)}(\tilde{s}') \\
&= \bar{b}_{\chi(h)}(\bar{s}') \cdot \sum_{\tilde{s}' \in \tilde{S}} \tilde{b}_{\xi(h)}(\tilde{s}') \\
&= \bar{b}_{\chi(h)}(\bar{s}')
\end{aligned}
$$

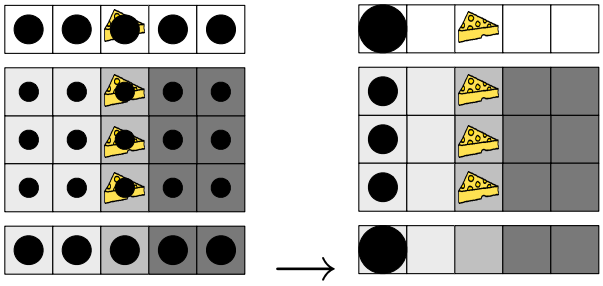
□

This concludes the proof that Procedure 2.4.1 succeeds only if κ, f and g form a valid POMDP Homomorphism.



(a) Initial Belief State (b) Belief conditioned on $\tilde{s}$

Figure 2.6. Independence test for the initial belief state test, domain from Figure 2.3. Black circles represent probability mass. The abstract shadow model is shown to the right of the gridworld, and the abstract model is shown below it.



(a) Initial Belief State (b) Belief conditioned on $\tilde{s}$

Figure 2.7. Independence test for the initial belief state test, domain from Figure 2.4. Black circles represent probability mass. The abstract shadow model is shown above the gridworld, and the abstract model is shown below it.

Figure 2.6 illustrates the first step of the independence test for the domain in Figure 2.3, in which the initial belief state is tested. The illustration, rather than looking at the marginals for the abstract and shadow states, examines the conditional for the abstract state given the shadow state. Expanding the notation, define $P(\bar{s} \mid b_\lambda) = \sum_{s \in \bar{s}} b_\lambda(s)$ and $P(\bar{s} \wedge \tilde{s} \mid b_\lambda) = \sum_{s \in \bar{s} \cap \tilde{s}} b_\lambda(s)$, etc. Since we know from probability theory that the following are equivalent tests:

$$P(\bar{s} \wedge \tilde{s} \mid b_\lambda) = P(\bar{s} \mid b_\lambda) \cdot P(\tilde{s} \mid b_\lambda) \iff P(\bar{s} \mid \tilde{s} \wedge b_\lambda) = P(\bar{s} \mid b_\lambda)$$

either test can be used to verify Equation 2.23.

Figure 2.6(a) illustrates the initial belief state in the model M (center), abstract model $\bar{M}_\chi$ (below M) and abstract shadow model $\tilde{M}_\xi$ (to the right of M). The probability masses for abstract and shadow states are calculated from the marginals of the belief state of M. Figure 2.6(b) illustrates the updated initial belief in all three models, conditioned on a specific abstract shadow state, $\tilde{s}$. Observation of the shadow state label narrows the belief distribution in the POMDP M. However, notice that the abstract state distribution does not change from Figure 2.6(a) to Figure 2.6(b). This indicates that for this shadow state in the initial belief distribution, the abstract model is independent. In order to prove that $\bar{M}_\chi$ and $\tilde{M}_\xi$ are independent, the full set of tests in Equations 2.23 and 2.24 must be verified.

Figure 2.7 illustrates one step of the initial belief state test in the domain from Figure 2.4. In this case, the states of the abstract and abstract shadow model are not independent. Figure 2.7(b) again illustrates the initial belief distribution in M, $\bar{M}_\chi$ and $\tilde{M}_\xi$. Figure 2.7(b) illustrates the updated belief state after conditioning on a particular abstract shadow state, $\tilde{s}$. In this case, revealing the abstract shadow state label improves the estimate of the abstract state label, and thus the abstract shadow model and abstract model are not independent.

2.4.6 Time Analysis

Procedure 2.4.1 is dominated by the time needed to construct the abstract model and abstract shadow model.

The CMP Homomorphism finding algorithm of Procedure 2.4.2, which is executed once to find the abstract model and once to find the abstract shadow model, has a worst case running time of $O(|S|^3 \cdot |A|)$. The outer while loop executes at most $|S|$ iterations, since the state mapping function f must change by at least one state on each iteratio. A tighter bound on the number of iterations would be $|\bar{S}|$ or $|\tilde{S}|$, the size of the abstract or abstract shadow state space, since each iteration must introduce at least one new abstract state. However, this is upper bounded by the number of states in S. Within each iteration, the algorithm

examines the abstract next state distribution for each state, action pair. The inner for loops iterate for $|S|$ and $|A|$ steps, respectively. One method of calculating the abstract next state distribution for a specific state, action pair is to iterate over the possible next states, adding the probability mass for each state to the probability mass of its abstract label. This process involves $|S|$ steps, as it examines each possible next state. The total worst case running time of the CMP Homomorphism finding algorithm is thus $O(|S| \cdot |S| \cdot |A| \cdot |S|)$, or $O(|S|^3 \cdot |A|)$.

The final independence test loop must calculate the joint distribution of the next abstract and shadow state labels. These calculations must be performed once for each state, action pair. The time needed to do one such calculation is $O(|S|)$ (to examine the labels for each possible next state and add its probability to the proper term). The entire loop therefore takes $O(|S| \cdot |A| \cdot |S|)$ in the worst case, and is dominated in this case by the CMP Homomorphism construction step.

2.4.7 Shortcomings of the Shadow Model

When every observation does not occur in every state, the shadow model algorithm does not perform well. Consider the gridworld in Figure 2.8, defined as the following POMDP:

States: Each square in Figure 2.8 represents a state.

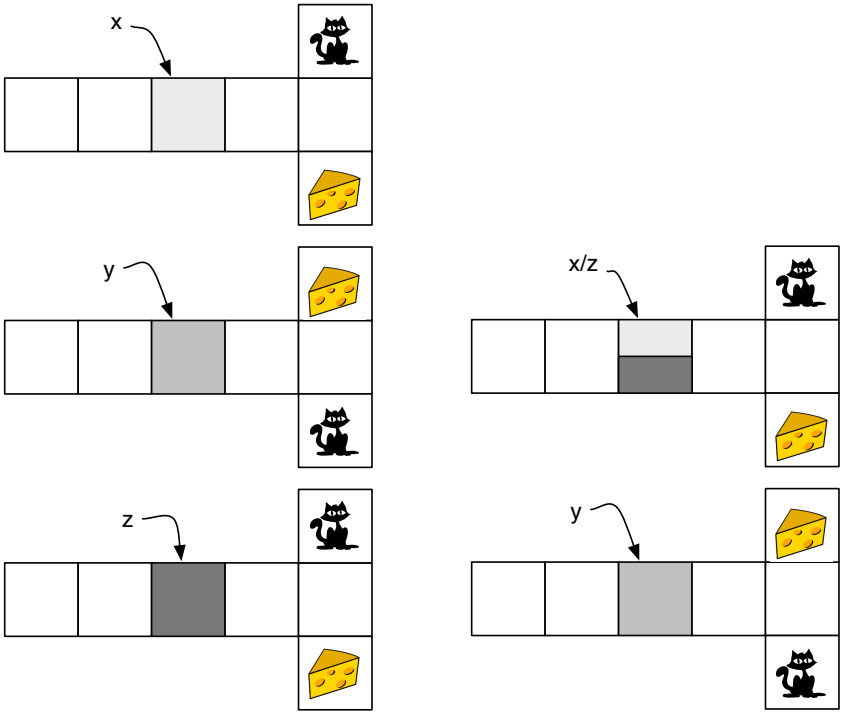
Actions: up, $down$, $left$, $right$

Transitions: Actions fail with a small probability ϵ. Failure results in no change to the state.

Observations: $white$, $lightgrey$, $grey$, $black$, $cheese$, cat

Observation Function: In each state, the agent *deterministically* observes only the features of the current square.

Initial Belief State: Equal probability mass on the leftmost state in each corridor.



(a) Three Hallway POMDP (b) Abstract Model

Figure 2.8. Three corridor gridworld POMDP. The initial state distribution places the agent in the leftmost state of each corridor with equal probability. The colors of the states labeled "x", "y" and"z" signal whether the agent must go straight or turn right at the end of the corridor to choose between the cheese and the cat.

The problem arises when $\frac{P(o|s,a)}{P(\kappa(o)|s,a)}$ is undefined, due to the fact that $P(\kappa(o) \mid s, a) = 0$. There are a few ways working around this by defining this term under these circumstances:

- treat *undefined* as a unique symbol ($\perp$)
- replace *undefined* with a small real ϵ probability
- replace *undefined* with 0.

However, none of these approaches really addresses the problem.

Define $\bar{O}$ as the set containing the following abstract observations:

$$\bar{o}_1 \stackrel{\text{def}}{=} \{cheese\}$$

$$\bar{o}_2 \stackrel{\text{def}}{=} \{cat\}$$

$$\bar{o}_3 \stackrel{\text{def}}{=} \{grey\}$$

$$\bar{o}_4 \stackrel{\text{def}}{=} \{black, lightgrey\}$$

$$\bar{o}_5 \stackrel{\text{def}}{=} \{white\}$$

The abstract model for this abstract observation function is shown in Figure 2.8(b). The shadow model state mapping function f_ξ is the identity function, so that the states of the abstract shadow model are the states of the original model, Figure 2.8(a). The two models do not pass the independence tests (Equations 2.23 and 2.24). However, this abstract observation function is in fact self sufficient. Part of the problem with the shadow model test for this domain lies in states x and y from Figure 2.8. State x has the following shadow observation distribution, for any action a:

$$cheese : \perp$$

$$cat : \perp$$

$$lightgrey : 1.0$$

$$grey : \perp$$

$$black : \perp$$

$$white : \perp$$

while state y has:

$$cheese : \perp$$

$$cat : \perp$$

$$lightgrey : \perp$$

$$grey : 1.0$$

$$black : \perp$$

$$white : \perp$$

Since these two shadow observation distributions are not the same, states x and y cannot have the same abstract shadow state label. Replacing $\perp$ with ϵ or 0 does not change this fact. In general, this should indicate that there is some discriminative power left in the observation distributions of x and y, but that is not the case here. Every distinction that can be made between these two states has been made, and the abstract observation function perfectly discriminates between these two states. The shadow model does not reflect this fact.

Even the observation map:

$$\bar{o}_1 \stackrel{\text{def}}{=} \{cheese\}$$

$$\bar{o}_2 \stackrel{\text{def}}{=} \{cat\}$$

$$\bar{o}_3 \stackrel{\text{def}}{=} \{lightgrey\}$$

$$\bar{o}_4 \stackrel{\text{def}}{=} \{grey\}$$

$$\bar{o}_5 \stackrel{\text{def}}{=} \{black\}$$

$$\bar{o}_6 \stackrel{\text{def}}{=} \{white\}$$

in which all observations distinctions available are used, does not pass the shadow model test. Similarly, if the domain in Figure 2.4 were altered to have deterministic observations, there would be no observation abstraction that passed the shadow model test.

It is to be expected that any approximation of Equation 2.12 would reject some valid abstractions. However, there is a large class of POMDPs for which the shadow model test does not accept any abstraction. The problem is that when the ratio $\frac{P(o|s,a)}{P(\kappa(o)|s,a)}$ is undefined, it should be treated as potentially equivalent to any other probability. The next algorithm treats undefined shadow observation probabilities as *compatible* with any other fixed observation probability.

2.5 Compatible Shadow States

Equations 2.23 and 2.24 require exact equivalence among the shadow states. In this section these requirements will be relaxed, and a *compatibility* relation over the shadow states will be constructed. The compatibility function is not an equivalence relation over the states, and cannot be used to construct a state mapping function.

The compatibility function takes advantage of the fact that if $\frac{P(o|s,a)}{P(\kappa(o)|s,a)}$ is not defined, $\kappa(o)$ will never be observed immediately after state s and action a. The ratio could therefore be reassigned any arbitrary value without affecting predictions. In other words, it does not matter what the conditional probability of o given $\kappa(o)$ is if $\kappa(o)$ will not occur. It is possible to find better abstractions if these values are treated as undefined in the sense that they could take on any value without affecting the model.

The state compatibility function is a boolean function $\sim_c$: $S \times S \rightarrow \{true, false\}$. Two states i and j are compatible if and only if their observation functions are compatible, and their next state distributions are compatible:

$$i \sim_c j \iff (i \sim_o j) \wedge (i \sim_t j)$$

where $\sim_o$: $S \times S \rightarrow \{true, false\}$ is an observation functions compatibility function, and $\sim_t$: $S \times S \rightarrow \{true, false\}$ is a next state distribution compatibility function.

The two states i and j have compatible observations (i.e. $i \sim_o j$) if and only if, for all $a \in A$ and $o \in O$:

$$((P(\kappa(o) \mid i, a) > 0) \wedge (P(\kappa(o) \mid j, a) > 0)) \rightarrow \frac{P(o \mid i, a)}{P(\kappa(o) \mid i, a)} = \frac{P(o \mid j, a)}{P(\kappa(o) \mid j, a)} \quad (2.25)$$

This means that the two states must have the same observation ratios, but only when $\kappa(o)$ is possible in both states. This relation is not an equivalence relation. Take states x, y and z in the domain from Figure 2.8, under the abstraction shown in Figure 2.8(b). The states x and y have compatible observation distributions ($x \sim_o y$), and y and z are compatible ($y \sim_o z$), however, x and z are not compatible. The observation compatibility function is not transitive, and therefore not an equivalence relation.

The next state distribution compatibility relation $\sim_t$ is defined recursively, in terms of the compatibility of the states in the two next state distributions. Two abstract POMDPs will be important in defining the relation $\sim_t$. One has already been defined: the candidate abstract POMDP $\bar{M}_\chi$ (Equation 2.15) with belief states $\bar{b}_{\chi(h)}$. The second abstract model is an abstract *availabily* POMDP. Define the availability function $\eta : \bar{O} \times S \times A \rightarrow \{0, 1\}$ as:

$$\eta(\bar{o}, s, a) = \begin{cases} 1 & \text{if } P(\bar{o} \mid s, a) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2.26)$$

This function indicates whether each abstract observation is available for a particular state and action.

The availability POMDP M_η is defined as the tuple

$$M_\eta = (S, A, T, \bar{O}, \eta), \quad (2.27)$$

This is not precisely a POMDP, as the observation probabilities do not sum to one in each state. Nonetheless, the usual POMDP update rule can be used to maintain a belief vector over the states. Define the state and action maps $f_\eta : S \rightarrow \bar{S}_\eta$ and $g_\eta : A \rightarrow \bar{A}_\eta$

as a CMP Homomorphism on (S, A, T), with output function $(\bar{O}, \Omega_\eta)$. The usual CMP Homomorphism properties apply, so that for any $s \in S$, $a \in A$ and $o \in O$:

$$\bar{\eta}(\bar{o}, f_\eta(s), g_\eta(a)) = \eta(\bar{o}, s, a) \qquad \text{from Equation 1.5} \qquad (2.28)$$

$$P(f_\eta(s') \mid f_\eta(s), g_\eta(a)) = \sum_{s'' \in [s']_{f_\eta}} P(s'' \mid s, a) \qquad \text{from Equation 1.6} \qquad (2.29)$$

Define the abstract availability model $\bar{M}_\eta$ as:

$$\bar{M}_\eta = (\bar{S}_\eta, \bar{A}_\eta, \bar{T}_\eta, \bar{O}, \bar{\eta}), \qquad (2.30)$$

where $\bar{T}$ is consistent with Equation 2.29 and $\bar{\eta}$ is consistent with Equation 2.28. Belief states for $\bar{M}_\eta$ will be written $\bar{b}_{\eta(h)}$.

The next state distributions for two states i and j are compatible ($i \sim_t j$) if and only if there exists a function for each action, $w_{ija} : S \times S \to I\!R$ that has the following three properties. If a pair k, l of next states are incompatible, they have zero weight:

$$\neg(k \sim_c l) \to (w_{ija}(k, l) = 0). \qquad (2.31)$$

For all abstract states $\bar{s}$ in $\bar{S}_\eta$ and any next state $k \in S$:

$$\sum_{l \in \bar{s}} w_{ija}(k, l) = P(k \mid i, a) \cdot P(\bar{s} \mid j, a). \qquad (2.32)$$

Finally, for all abstract availability states $\bar{s}$ in $\bar{S}$, for any next state $l \in S$:

$$\sum_{k \in \bar{s}} w_{ija}(k, l) = P(\bar{s} \mid i, a) \cdot P(l \mid j, a). \qquad (2.33)$$

Reflexivity and symmetry may be violated by the constraints in Equations 2.31 - 2.33. It is possible that for some state i there is no weight function w_{iia}, therefore the function is

not reflexive. As for symmetry, Equations 2.32 and 2.33 are similar, however, the abstract mapping for state k is f and the abstract mapping for state l is f_η and the existence of w_{ija} does not imply the existence of w_{jia}. The next state compatibility function is thus not an equivalence relation.

However the compatibility function $\sim_c$ can be used to inspect $\bar{M}_\chi$ for correctness. For the compatibility test to pass, the initial state distribution b_λ must be compatible with itself. That is, there must be a weight function $w_\lambda : S \times S \rightarrow I\!R$ such that for all $i, j \in S$:

$$\neg(i \sim_c j) \rightarrow (w_\lambda(i,j) = 0) \tag{2.34}$$

For all abstract states $\bar{s}$ in $\bar{S}_\eta$:

$$\sum_{j \in \bar{s}} w_\lambda(i,j) = b_\lambda(i) \cdot b_{\eta(\lambda)}(\bar{s}) \tag{2.35}$$

And for all abstract availability states $\bar{s}$ in $\bar{S}$:

$$\sum_{i \in \bar{s}} w_\lambda(i,j) = b_{\chi(\lambda)}(\bar{s}) \cdot b_\lambda(j) \tag{2.36}$$

For example, if the states of Figure 2.8 are labeled as shown in Figure 2.9, then the state compatibility function for the three states in the initial state distribution (s_0, s_7 and s_{14}) is:

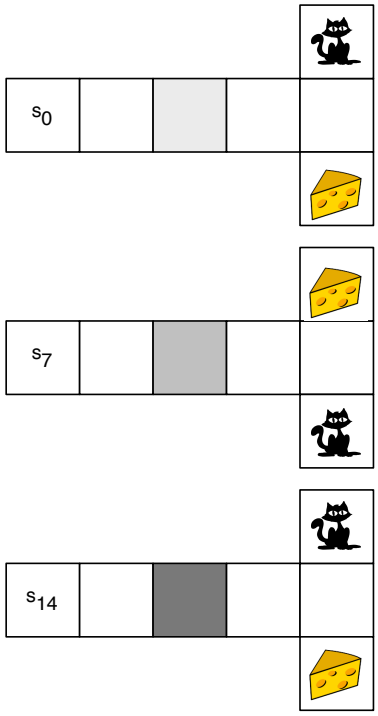


Figure 2.9. Three corridor gridworld POMDP from Figure 2.8, with starting state labels. The initial state distribution places the agent in s_0, s_7, and s_{14} with equal probability.

$$s_0 \sim_c s_0 = true$$

$$s_0 \sim_c s_7 = true$$

$$s_0 \sim_c s_{14} = false$$

$$s_7 \sim_c s_0 = true$$

$$s_7 \sim_c s_7 = true$$

$$s_7 \sim_c s_{14} = true$$

$$s_{14} \sim_c s_0 = false$$

$$s_{14} \sim_c s_7 = true$$

$$s_{14} \sim_c s_{14} = true$$

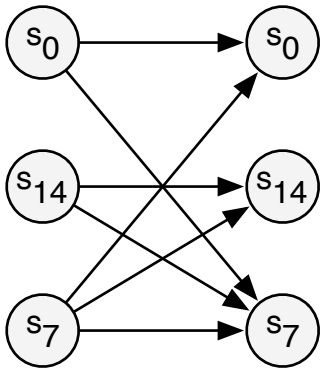
and the initial belief matching function is:

$$w_\lambda(s_0, s_0) = \frac{2}{9}$$
$$w_\lambda(s_0, s_7) = \frac{1}{9}$$

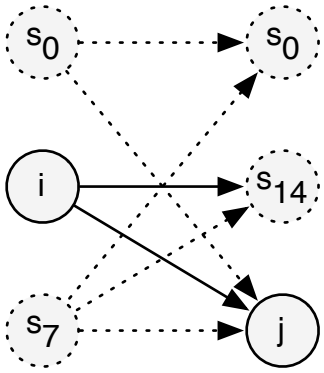
$$w_\lambda(s_7, s_0) = \frac{1}{9}$$
$$w_\lambda(s_7, s_7) = \frac{1}{9}$$
$$w_\lambda(s_7, s_{14}) = \frac{1}{9}$$

$$w_\lambda(s_{14}, s_{14}) = \frac{2}{9}$$
$$w_\lambda(s_{14}, s_7) = \frac{1}{9}$$

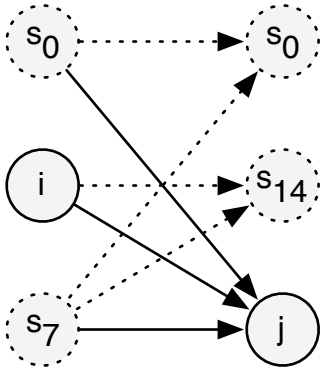
with all state pairs not listed having 0 weight.



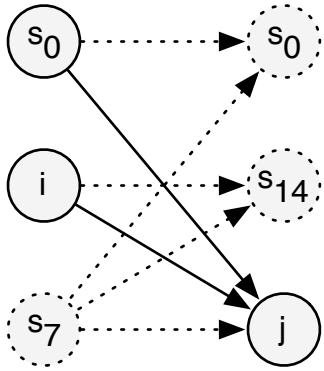
(a) Compatible Edges



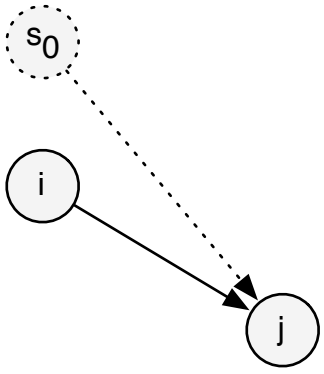
(b) out(i)



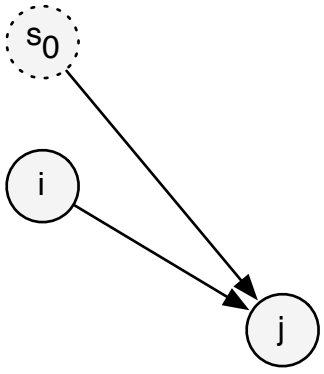
(c) in(j)



(d) Abstract i-j graph



(e) Abstract out(i)



(f) Abstract in(j)

Figure 2.10. The function w_λ for the domain of Figure 2.9 for the abstraction shown in Figure 2.8(b), illustrated as a graph

The initial belief matching function can be visualized as a set of edges in a weighted graph, illustrated in Figure 2.10. Figure 2.10(a) includes two nodes for each state in b_λ: a left and right node. Left to right edges are present only between pairs where the left node state is compatible with the right node state. So, for example, there is no edge between s_0 and s_{14}, since s_0 and s_{14} are not compatible. Each edge between two nodes i and j is weighted according to $w_\lambda(i,j)$, and these weights have several useful properties. The sum of the out edges on any left node, like the node i in Figure 2.10(b), is equal to $b_\lambda(i)$. Similarly, the sum of the weights of the incoming edges for any right node j is equal to $b_{\eta(\lambda)}(j)$.

Consider the subgraph consisting of left nodes that share the abstract label $f(i)$, and right nodes that share the abstract label $f_\eta(j)$, highlighted in Figure 2.10(d). Equations 2.35 and 2.36 also imply that the in and out weight totals in this subgraph have interesting properties. The sum of the outgoing edges from i in this subgraph (Figure 2.10(e)) is given in Equation 2.35, and the sum of the ingoing edges to j in this subgraph (Figure 2.10(f)) is given in Equation 2.36.

Equations 2.34 - 2.36 are the compatibility test constraints. The next section consists of a proof that if the compatibility test constraints are satisfied, and $\sim_c$ is a compatibility function satisfying Equations 2.25 and 2.31 - 2.33, then f, g, κ is a POMDP Homomorphism. The following section details an algorithm for calculating the compatibility function and w_λ, implementing a polynomial time algorithm for checking these constraints.

2.5.1 Composite Model

This section proves that when Equations 2.34 - 2.36 are satisfied, $\bar{M}_\chi$ is a valid homomorphic reduction of M. The proof examines a composite POMDP denoted $\breve{M}$. This section will outline the structure of $\breve{M}$, show that it could be constructed if the compatibility properties hold, and show that the fact of its existence implies the correctness of $\bar{M}$. It

should be noted that $\breve{M}$ never needs to actually be constructed. Its theoretical existence, given Equations 2.34 - 2.36, is sufficient to prove the correctness of $\bar{M}$.

The states of $\breve{M}$ each consist of a pair of state labels from M: for example, $\langle i, j \rangle$, where $i, j \in S$. $\breve{M}$ will be structured such that only compatible state pairs are used.

The left hand state labels in $\breve{M}$ emit abstract observations, and the right hand state labels emit "shadow" observations. That is:

$$P(o \mid \langle i,j \rangle) = \overbrace{P(\kappa(o) \mid i,a)}^{\text{abstract observation}} \cdot \overbrace{\frac{P(o \mid j,a)}{P(\kappa(o) \mid j,a)} \cdot \eta(\kappa(o),j,a)}^{\text{shadow observation}} \tag{2.37}$$

Recall that the function $\eta : \bar{O} \times S \times A \rightarrow \{0,1\}$ (Equation 2.26) indicates whether the abstract observation is available for a particular state and action. This has the effect of setting the shadow observation probabilities to 0 whenever $\frac{P(o|j,a)}{P(\kappa(o)|j,a)}$ is undefined.

The transition function $\breve{T}$ for any compatible state pair $\langle i, j \rangle$ and action a is defined via the weight of their next state transition compatibility matching w_{ija}, and the abstract state predictions for i and j:

$$P(\langle k,l \rangle \mid \langle i,j \rangle, a) = w_{ija}(k,l) \tag{2.38}$$

The initial belief state for $\breve{M}$ is defined by the compatibility weights for b_λ:

$$\hat{b}_\lambda(\langle i,j \rangle) = w_\lambda(i,j) \tag{2.39}$$

Since w_{ija} and w_λ are always 0 for pairs of incompatible states, neither the initial belief vector nor the transition function will introduce incompatible state pairs into the belief vector.

The belief state update rule is:

$$\breve{b}_{hao}(\langle i,j\rangle) = \frac{P(o \mid \langle i,j\rangle, a) \cdot \breve{b}_{ha}((\langle i,j\rangle)}{\sum_{i,j} P(o \mid \langle i,j\rangle, a) \cdot \breve{b}_{ha}(\langle i,j\rangle)}$$

Strictly speaking $\breve{M}$ is not a POMDP, as the observation output probabilities defined in Equation 2.37 do not necessarily sum to one after each belief state update. This can be remedied by introducing an additional normalization constant at each history hao:

$$c_{hao} = \sum_{\bar{s} \in \bar{S}_\eta} \bar{\eta}(\kappa(o), \bar{s}, g_\eta(a)) \cdot \bar{b}_{\eta(ha)}(\bar{s})$$

which must be applied to recover the correct observation probabilities for the belief state $\breve{b}_{ha}$. The probability of o given $\breve{b}_{ha}$ is:

$$P(o \mid \breve{b}_{ha}) = \frac{\sum_{i,j} P(o \mid \langle i,j\rangle, a) \cdot \breve{b}_{ha}(\langle i,j\rangle)}{c_{hao}}$$

There are several elemental identities that will be used as building blocks for most of the main proofs in this section. If i and j are two compatible states:

$$\begin{aligned} P(o \mid \langle i,j\rangle) &= P(\kappa(o) \mid i,a) \cdot \frac{P(o \mid j,a)}{P(\kappa(o) \mid j,a)} \cdot \eta(\kappa(o), j, a) \\ &= P(\kappa(o) \mid i,a) \cdot \frac{P(o \mid i,a)}{P(\kappa(o) \mid i,a)} \cdot \eta(\kappa(o), j, a) \\ &= P(o \mid i,a) \cdot \eta(\kappa(o), j, a) \end{aligned} \tag{2.40}$$

Similarly, due to the properties of w_{ija} for two compatible states (Equations 2.32 and 2.33), if $\bar{s}_l \in \bar{S}, \bar{s}_r \in \bar{S}_\eta$:

$$\sum_{l \in \bar{s}_r} P(\langle k,l\rangle \mid \langle i,j\rangle, a) = P(k \mid i,a) \cdot P(\bar{s}_r \mid j,a) \tag{2.41}$$

$$\sum_{k \in \bar{s}_l} P(\langle k,l\rangle \mid \langle i,j\rangle, a) = P(\bar{s}_l \mid i,a) \cdot P(l \mid j,a) \tag{2.42}$$

$\breve{M}$ is well defined, and it will be shown to accurately simulate M. This can be used to show that $\bar{M}$ is a homomorphic reduction of M, by showing that:

- M and $\breve{M}$ are output-equivalent.
- $\bar{M}$ is a homomorphic reduction of $\breve{M}$.

Taken together, these two properties imply that $\bar{M}$ is a homomorphic reduction of M.

In order to analyze $\breve{M}$, the shadow POMDP M_ξ (Equations 2.19) will be useful, with one modification. The shadow observation function for any state s, action a and observation o is now:

$$\Omega_\xi(s,a,o) = \frac{P(o \mid s,a)}{P(\kappa(o) \mid s,a)} \cdot \eta(\kappa(o),s,a) \tag{2.43}$$

where the availability function η has the effect of setting the shadow observation probabilities to 0 whenever $\frac{P(o|s,a)}{P(\kappa(o)|s,a)}$ is undefined.

The next theorem (Theorem 2.7) is the central theorem about $\breve{M}$.

Theorem 2.7. *Given Equations 2.31 - 2.36, for all h in H_M, the weight function w_h:*

$$w_h(i,j) = \breve{b}_h(i,j) \tag{2.44}$$

defined by $\breve{b}_h$ corresponds to a compatibility matching for the two state distributions b_h and $b_{\xi(h)}$. That is:

$$\neg(i \sim_c j) \rightarrow \breve{b}_h(i,j) = 0 \tag{2.45}$$

$$\sum_{j \in \bar{s}_r} \breve{b}_h(\langle i,j \rangle) = b_h(i) \cdot \bar{b}_{\eta(h)}(\bar{s}_r) \tag{2.46}$$

$$\sum_{i \in \bar{s}_l} \breve{b}_h(\langle i,j \rangle) = \bar{b}_{\chi(h)}(\bar{s}_l) \cdot b_{\xi(h)}(j) \tag{2.47}$$

for every $\bar{s}_r \in \bar{S}_\eta$ and $\bar{s}_l \in \bar{S}$.

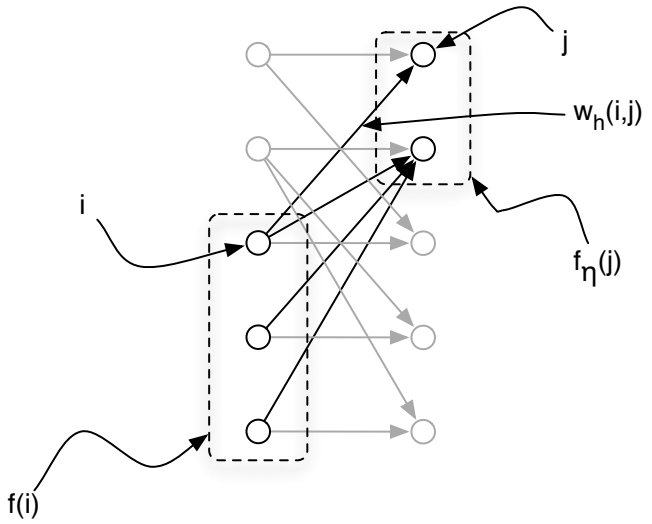


Figure 2.11. Matching graph for the belief state $\breve{b}_h$.

Before addressing the proof of Theorem 2.7, a high level overview of its implications and an outline of the proof.

Theorem 2.7 states that for every $h \in H_M$, there is a weight matching for the two distributions b_h and $b_{\xi(h)}$, and this weight function w_h can be derived from the belief state of $\breve{M}$, $\breve{b}_h$. Figure 2.11 illustrates the structure of $\breve{b}_h$ as a matching graph, where each edge corresponds to a state pair, weighted according to w_h.

Define the left hand state distribution $\breve{l}_h$ of a belief state $\breve{b}_h$, for any state $i \in S$ as:

$$\breve{l}_h(i) = \sum_{j \in S} \breve{b}_h(\langle i, j \rangle)$$

and the right hand state distribution $\breve{r}_h$ for any state $j \in S$ as:

$$\breve{r}_h(j) = \sum_{i \in S} \breve{b}_h(\langle i, j \rangle).$$

One of the implications of Theorem 2.7 is that the left state labels of $\breve{b}_h$ track the belief state of M, so that $\breve{l}_h = b_h$ (see Lemma 2.9). This fact implies that $\breve{M}$ is equivalent to

M in terms of its observation predictions (see Lemma 2.10). Theorem 2.7 further implies that the abstract belief state $\bar{b}_{\chi(h)}$ accurately tracks the left hand belief state $\breve{l}_h$, at the level of abstract state labels. Since $\breve{l}_h$ is equivalent to b_h, $\bar{b}_{\chi(h)}$ is then shown to be an accurate abstraction of b_h as well (see Theorem 2.11).

Theorem 2.7 also implies that the right hand distribution of $\breve{b}_h$ accurately tracks the shadow state ($\breve{r}_h = b_{\xi(h)}$), and that $\bar{b}_{\eta(h)}$ is an accurate abstract compression of $b_{\xi(h)}$. Although this fact is tangential to the main point of this section and the proof is not included here, it can be helpful in understanding the proofs.

Lemma 2.8. *Given Equations 2.31 - 2.36, for any $\bar{s}_l \in \bar{S}$ and $\bar{s}_r \in \bar{S}_\eta$: The action update belief state $\breve{b}_{ha}$ is also a weight matching:*

$$\neg(i \sim_c j) \rightarrow \breve{b}_{ha}(i,j) = 0 \tag{2.48}$$

$$\sum_{j \in \bar{s}_r} \breve{b}_{ha}(\langle i,j \rangle) = b_{ha}(i) \cdot \bar{b}_{\eta(ha)}(\bar{s}_r) \tag{2.49}$$

$$\sum_{i \in \bar{s}_l} \breve{b}_{ha}(\langle i,j \rangle) = \bar{b}_{\chi(ha)}(\bar{s}_l) \cdot b_{\xi(ha)}(j) \tag{2.50}$$

for every $\bar{s}_r \in \bar{S}_\eta$ and $\bar{s}_l \in \bar{S}$.

Proof. The proof of both Theorem 2.7 and Lemma 2.8 consists of a structural induction proof on historiesin H_M, with two interlocking inductive steps: one for the action update, and one for the observation update.

Part I: The base case: $h = \lambda$.

For the initial history λ, the due to the form of w_λ (Equations 2.35 and 2.36), for $\bar{s}_r \in \bar{S}_\eta$ and $\bar{s}_l \in \bar{S}$:

$$\sum_{j \in \bar{s}_r} \breve{b}_\lambda(\langle i, j \rangle) = \sum_{j \in \bar{s}_r} w_\lambda(i, j)$$
$$= b_\lambda(i) \cdot \bar{b}_{\eta(\lambda)}(\bar{s}_r)$$
$$\sum_{i \in \bar{s}_l} \breve{b}_\lambda(\langle i, j \rangle) = \sum_{i \in \bar{s}_l} w_\lambda(i, j)$$
$$= \bar{b}_{\chi(\lambda)}(\bar{s}_l) \cdot b_\lambda(j).$$

Part II: h to ha inductive step

Proof of Equation 2.49 assuming Equation 2.46. For $\bar{s}_r \in \bar{S}_\eta$:

$$\sum_{j \in \bar{s}_r} \breve{b}_{ha}(\langle i, j \rangle) = \sum_{j \in \bar{s}_r} \sum_{k,l \in S} P(\langle i, j \rangle | \langle k, l \rangle, a) \cdot \breve{b}_h(\langle k, l \rangle)$$
$$= \sum_{k,l \in S} P(i|k, a) \cdot P(\bar{s}_r|l, a) \cdot \breve{b}_h(\langle k, l \rangle) \quad \text{Equation 2.41}$$
$$= \sum_{k \in S} P(i|k, a) \cdot \sum_{\bar{s} \in \bar{S}_\eta} P(\bar{s}_r|\bar{s}, g(a)) \cdot \sum_{l \in \bar{s}} \breve{b}_h(\langle k, l \rangle) \quad \text{Equation 1.6}$$
$$= \sum_{k \in S} P(i|k, a) \cdot b_h(k) \sum_{\bar{s} \in \bar{S}_\eta} P(\bar{s}_r|\bar{s}, g(a)) \cdot \bar{b}_{\eta(h)}(\bar{s}) \quad \text{Equation 2.46}$$
$$= b_{ha}(i) \cdot \bar{b}_{\eta(ha)}(\bar{s}_r)$$

Proof of Equation 2.50 assuming Equation 2.47. For $\bar{s}_l \in \bar{S}$:

$$\sum_{i \in \bar{s}_l} \breve{b}_{ha}(\langle i, j \rangle) = \sum_{i \in \bar{s}_l} \sum_{k,l \in S} P(\langle i, j \rangle | \langle k, l \rangle, a) \cdot \breve{b}_h(\langle k, l \rangle)$$
$$= \sum_{k,l \in S} P(j|l, a) \cdot P(\bar{s}_l|k, a) \cdot \breve{b}_h(\langle k, l \rangle) \quad \text{Equation 2.42}$$
$$= \sum_{l \in S} P(j|l, a) \cdot \sum_{\bar{s} \in \bar{S}} P(\bar{s}_l|\bar{s}, g(a)) \cdot \sum_{k \in \bar{s}} \breve{b}_h(\langle k, l \rangle) \quad \text{Equation 2.11}$$
$$= \sum_{l \in S} P(j|l, a) \cdot b_h(l) \sum_{\bar{s} \in \bar{S}} P(\bar{s}_l|\bar{s}, g(a)) \cdot \bar{b}_{\chi(h)}(\bar{s}) \quad \text{Equation 2.47}$$
$$= b_{ha}(j) \cdot \bar{b}_{\chi(ha)}(\bar{s}_l)$$

Part III: ha to hao inductive step

Proof of Equation 2.46 assuming Equation 2.49:

$$\begin{aligned}
\sum_{j\in\bar{s}_r}\breve{b}_{hao}(\langle i,j\rangle) &= \sum_{j\in\bar{s}_r}\frac{P(o\mid\langle i,j\rangle,a)\cdot\breve{b}_{ha}(\langle i,j\rangle)}{\sum_{i,j\in S}P(o\mid\langle i,j\rangle,a)\cdot\breve{b}_{ha}(\langle i,j\rangle)}\\
&= \frac{\sum_{j\in\bar{s}_r}P(o\mid i,a)\cdot\eta(\kappa(o),j,a)\cdot\breve{b}_{ha}(\langle i,j\rangle)}{\sum_{i\in S}P(o\mid i,a)\cdot\sum_{\bar{s}\in\bar{S}_\eta}\sum_{j\in\bar{s}}\eta(\kappa(o),j,a)\cdot\breve{b}_{ha}(\langle i,j\rangle)} && \text{Equation 2.40}\\
&= \frac{P(o\mid i,a)\cdot\bar{\eta}(\kappa(o),\bar{s}_r,g(a))\cdot\sum_{j\in\bar{s}_r}\breve{b}_{ha}(\langle i,j\rangle)}{\sum_{i\in S}P(o\mid i,a)\cdot\sum_{\bar{s}\in\bar{S}_\eta}\bar{\eta}(\kappa(o),\bar{s},g(a))\cdot\sum_{j\in\bar{s}}\breve{b}_{ha}(\langle i,j\rangle)} && \text{Equation 2.28}\\
&= \frac{P(o\mid i,a)\cdot\bar{\eta}(\kappa(o),\bar{s},g(a))\cdot b_{ha}(i)\cdot\bar{b}_{\eta(ha)}(\bar{s})}{\sum_{i\in S}P(o\mid i,a)\cdot\sum_{\bar{s}\in\bar{S}_\eta}\bar{\eta}(\kappa(o),\bar{s},g(a))\cdot b_{ha}(i)\cdot\bar{b}_{\eta(ha)}(\bar{s})} && \text{Equation 2.49}\\
&= \frac{P(o\mid i,a)\cdot b_{ha}(i)\cdot\bar{\eta}(\kappa(o),\bar{s},g(a))\cdot\bar{b}_{\eta(ha)}(\bar{s})}{\sum_{i\in S}P(o\mid i,a)\cdot b_{ha}(i)\cdot\sum_{\bar{s}\in\bar{S}_\eta}\bar{\eta}(\kappa(o),\bar{s},g(a))\cdot\bar{b}_{\eta(ha)}(\bar{s})}\\
&= b_{hao}(i)\cdot\bar{b}_{\eta(hao)}(\bar{s})
\end{aligned}$$

Proof of Equation 2.50 assuming Equation 2.47[2]:

$$\begin{aligned}
\sum_{i\in\bar{s}_l}\breve{b}_{hao}(\langle i,j\rangle) &\propto \sum_{i\in\bar{s}_l}P(o\mid\langle i,j\rangle,a)\cdot\breve{b}_{ha}(\langle i,j\rangle)\\
&\propto \frac{P(o\mid j,a)}{P(\kappa(o)\mid j,a)}\eta(\kappa(o),j,a)\cdot P(\kappa(o)|\bar{s}_l,g(a))\cdot\sum_{i\in\bar{s}_l}\breve{b}_{ha}(\langle i,j\rangle)\\
&\propto \frac{P(o\mid j,a)}{P(\kappa(o)\mid j,a)}\eta(\kappa(o),j,a)\cdot P(\kappa(o)\mid\bar{s}_l,g(a))\cdot b_{\xi(h)}(j)\cdot\bar{b}_{\chi(h)}(\bar{s}_l)\\
&\propto \frac{P(o\mid j,a)}{P(\kappa(o)\mid j,a)}\eta(\kappa(o),j,a)\cdot b_{\xi(h)}(j)\cdot P(\kappa(o)\mid\bar{s}_l,g(a))\cdot\bar{b}_{\chi(h)}(\bar{s}_l)\\
&\propto b_{\xi(hao)}(j)\cdot\bar{b}_{\chi(hao)}(\bar{s}_l)
\end{aligned}$$

□

This ends the proof of Lemma 2.8 and Theorem 2.7.

[2] I am showing just the numerator in this proof due to the length of the equations. The transformation for the denominator is similar, and factors into $\sum_j b_{\xi(hao)}(j)\cdot\sum_{\bar{s}_l}\bar{b}_{\chi(hao)}(\bar{s}_l)$.

The next several Lemmas show that given Theorem 2.7 and Lemma 2.8, $\breve{M}$ and M are output equivalent, and that $\bar{M}$ is a homomorphic reduction of M.

Lemma 2.9. *If $\breve{M}$ satisfies Theorem 2.7, the state distribution of the left side labels for $\breve{M}$ is the same as the state distribution of the belief state in M. That is:*

$$\forall h \in H_M, \breve{l}_h(i) = b_h(i)$$

Proof.

$$\begin{aligned}
\breve{l}_h(i) &= \sum_{j \in S} \breve{b}_h(\langle i, j \rangle) && \text{by definition} \\
&= \sum_{\bar{s}_r \in \bar{S}} \sum_{j \in \bar{s}_r} \breve{b}_h(\langle i, j \rangle) \\
&= b_h(i) \cdot \sum_{\bar{s}_r \in \bar{S}} \bar{b}_{\eta(h)}(\bar{s}_r) && \text{Equation 2.46} \\
&= b_h(i) && b_{\eta(h)} \text{ belief state distribution sums to 1}
\end{aligned}$$

□

Lemma 2.10. *If $\hat{M}$ satisfies Lemma 2.8, the observation distribution at every history ha is identical to the observation distribution for the left side state labels, and thus for the corresponding belief state in M.*

Proof. For all $o \in O$, $a \in A$ and $h \in H_M$,

$$
\begin{aligned}
P(o \mid \breve{b}_{ha}) &= \frac{\sum_{i,j \in S} P(o \mid \langle i,j \rangle, a) \cdot \breve{b}_{ha}(\langle i,j \rangle)}{c_{hao}} && \text{by definition} \\
&= \frac{\sum_{i,j \in S} P(o|i,a) \cdot \eta(\kappa(o), j, a) \cdot \breve{b}_{ha}(\langle i,j \rangle)}{c_{hao}} && \text{Equation 2.40} \\
&= \frac{\sum_{i \in S} P(o|i,a) \cdot \sum_{\bar{s} \in \bar{S}_\eta} \bar{\eta}(\kappa(o), \bar{s}, g_\eta(a)) \cdot \sum_{j \in \bar{s}} \breve{b}_{ha}(\langle i,j \rangle)}{c_{hao}} \\
&= \frac{\sum_{i \in S} P(o|i,a) \cdot b_{ha}(i) \cdot \sum_{\bar{s} \in \bar{S}_\eta} \bar{\eta}(\kappa(o), \bar{s}, g_\eta(a)) \cdot \bar{b}_{\eta(ha)}(\bar{s})}{c_{hao}} && \text{Equation 2.49} \\
&= \sum_{i \in S} P(o|i,a) \cdot b_{ha}(i) && \text{Cancel terms} \\
&= P(o \mid b_{ha}).
\end{aligned}
$$

□

This concludes the proof that $\breve{M}$ simulates M accurately. Now this fact can be used to show that f, g and κ satisfy the homomorphism constraints for M: Equations 2.9 - 2.12. Equation 2.9 is checked directly in Procedure 2.5.1. Equations 2.10 and 2.11 are satisfied because f and g form a CMP Homomorphism for $\bar{O}$. All that remains is to show that Equation 2.12 is satisfied.

Theorem 2.11. *If $\breve{M}$ satisfies Theorem 2.7, then for any $\bar{s} \in \bar{S}$, for all $h \in H_M$, $\sum_{i \in \bar{s}} b_h(i) = \bar{b}_{\chi(h)}(\bar{s})$ (Equation 2.12).*

Proof.

$$
\begin{aligned}
\sum_{i \in \bar{s}} b_h(i) &= \sum_{i \in \bar{s}} \breve{l}_h(i) && \text{Lemma 2.9} \\
&= \sum_{i \in \bar{s}} \sum_{j \in S} \breve{b}_h(\langle i,j \rangle) \\
&= \bar{b}_{\chi(h)}(\bar{s}) \cdot \sum_{j \in S} b_{\xi(h)}(j) && \text{Theorem 2.7} \\
&= \bar{b}_{\chi(h)}(\bar{s}) && b_{\xi(h)} \text{ sums to 1.}
\end{aligned}
$$

□

Procedure 2.5.1 State Compatibility Check(M, κ)

// Check that κ satisfies Equation 2.9 ($\zeta(o, z) = \bar{\zeta}(\kappa(o), z)$)
// Construct the abstract model $\bar{M}_\chi$ and state and action mapping functions f, g
$f, g \Leftarrow$ findCMPHomomorphism($S, A, T, output = \bar{O}, \Omega_\chi$)

// Construct availability model $\bar{M}_\eta$ and state and action mapping functions f_η, g_η
$f_\eta, g_\eta \Leftarrow$ findCMPHomomorphism($S, A, T, output = \bar{O}, \Omega_\eta$)

// Construct the compatibility function
compatibleStates $\Leftarrow$ constructCompatibilityFunction(M, κ, f, f_η)

// Do final check, given compatibility among shadow states
if compatibleDistributions($b_\lambda, b_\lambda, f, f_\eta$, compatibleStates) **then**
  return true
else
  return false

This concludes the proof that the state compatibility algorithm (Procedure 2.5.1) succeeds only if $\bar{M}$ is a homomorphic reduction of M. The algorithm accurately detects correct and invalid homomorphisms in the examples in the next section, however, it has not been proven to be complete. There are cases in which this algorithm would reject a valid homomorphism.

2.5.2 Compatibility Algorithm

This section will show that the compatibility test can be computed in polynomial time. The state compatibility algorithm (Procedure 2.5.1) first constructs $\bar{M}_\chi$ and $\bar{M}_\eta$ with their associated state and action mapping functions. Next, the algorithm constructs the compatibility function (Procedure 2.5.2), and checks the initial belief state b_λ for compatibility.‘

Procedure 2.5.2, which calculates the compatibility function, first initializes the compatibility function based on the immediate observation compatibility of the states. This is a direct check of Equation 2.25. Then the main loop of the procedure repeatedly refines the compatibility function to ensure that the transition constraints, Equations 2.31 - 2.33 are satisfied. In the worst case, this loop halts when all states have been declared incompatible.

Procedure 2.5.2 constructCompatibilityFunction(M, κ, f, f_η)

```
compatibleStates_0 ⇐ |S| × |S| boolean matrix
// Initialize the compatibility function using the observation function
for all i, j ∈ S do
  compatibleStates(i, j) ⇐ observationsCompatible(i, j) // Check Equation 2.25

// Refine the compatibility function until compatible states have compatible next state distributions
repeat
  compatibleStatesOld ⇐ compatibleStates
  for all i, j ∈ S, a ∈ A do
    if ¬compatibleDistributions(P(S|i, a), P(S|j, a), f, f_η, compatibleStates) then
      compatibleStates(i, j) ⇐ false
until compatibleStatesOld = compatibleStates

return compatibleStates
```

Procedure 2.5.3 compatibleDistributions($P_L(S), P_R(S), f, f_\eta, compatibleStates$)

```
for s̄_l ∈ S̄ do
  for s̄_r ∈ S̄_η do
    graph ⇐ constructDistributionGraph(P_L, s̄_l, P_R, s̄_r, compatibleStates)
    flow ⇐ maxFlow(graph)
    if flow(graph.source) < 1 then
      return false
return true
```

Procedure 2.5.3 is key to the implementation of both Procedure 2.5.1 and 2.5.2. This subroutine checks two state distributions to determine whether they are compatible given f, f_η and the current compatibility function. The two state distributions P_L and P_R may be:

- the next state distributions for two states i and j under action a, in which case the procedure must construct w_{ija} or return failure.
- two copies of the initial state distribution b_λ, in which case the procedure must construct w_λ or return failure.

In both cases, the weight function w_{ija} or w_λ can be constructed as the sum of a set of graph flow weights. The set consists of a flow graph for each pair of abstract states $\bar{s}_l$ from

Procedure 2.5.4 constructDistributionGraph($P_L(S), \bar{s}_l, P_R(S), \bar{s}_r, compatibleStates$)

$V \Leftarrow \{s, t\}$ // new set of vertices, source and sink vertex
$capacity : V \times V \rightarrow I\!R$ // Edge capacity function

// Fill in the nodes and edges of the graph
for all $i \in \bar{s}_l$ **do**
  $V \Leftarrow V \cup \{l_i\}$
  $capacity(s, l_i) = \frac{P_L(i)}{\sum_{i' \in \bar{s}_l} P_L(i')}$
for all $j \in \bar{s}_r$ **do**
  $V \Leftarrow V \cup \{r_j\}$
  $capacity(r_j, t) = \frac{P_R(j)}{\sum_{j' \in \bar{s}_r} P_R(j')}$
for all $i \in \bar{s}_l$**,** $j \in \bar{s}_r$ **do**
  if $compatibleStates(i, j)$ **then**
    $capacity(l_i, r_j) = 1$
  else
    $capacity(l_i, r_j) = 0$

return $graph = (V, capacity)$

$\bar{S}$ and $\bar{s}_r$ from $\bar{S}_\eta$. A flow graph for $\bar{s}_l$ and $\bar{s}_r$ is shown in Figure 2.12. The vertices of the graph are:

- s (source node)
- $L = \{l_i \mid i \in \bar{s}_l\}$ (state nodes in $\bar{s}_l$)
- t (sink node)
- $R = \{r_j \mid j \in \bar{s}_r\}$ (state nodes in $\bar{s}_r$)

If $cap(u, v)$ is the edge capacity between node u and node v:

$$cap(s, l_i) = \frac{P_L(i)}{\sum_{i' \in \bar{s}_l} P_L(i')} \tag{2.51}$$

$$cap(l_i, r_j) = \begin{cases} 1 & \text{if } i \sim_c j \\ 0 & \text{otherwise} \end{cases} \tag{2.52}$$

$$cap(r_j, t) = \frac{P_R(j)}{\sum_{j' \in \bar{s}_r} P_R(j')} \tag{2.53}$$

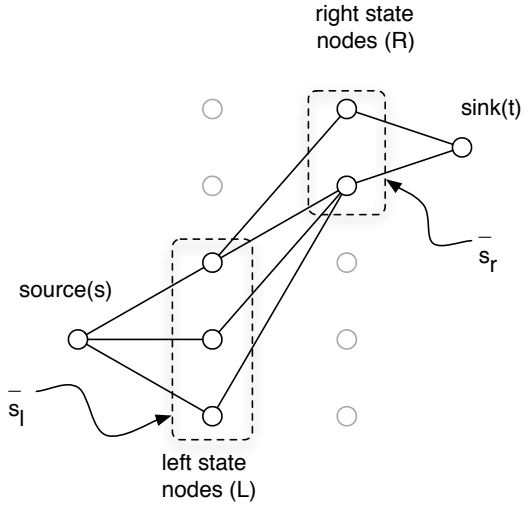
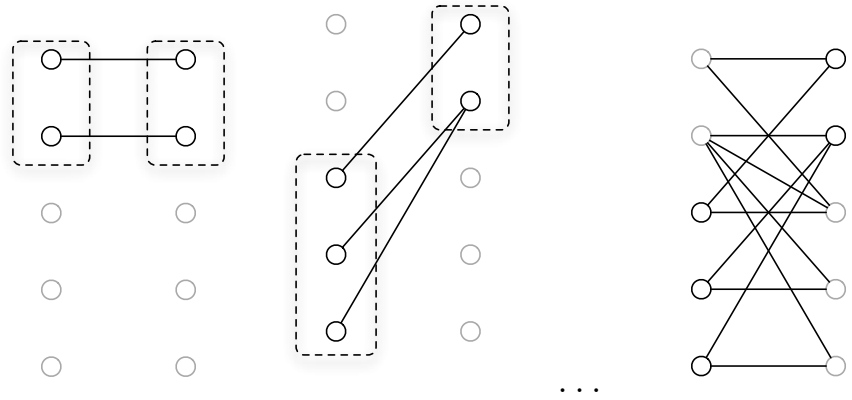


Figure 2.12. Matching algorithm graph for the abstract states $\bar{s}_l$ (in the left side distribution) and $\bar{s}_r$ (in the right side distribution). See the text for edge weight definitions.



(a) Edges used for $\bar{s}_0, \bar{s}_0$

(b) Edges used for $\bar{s}_1, \bar{s}_0$

(c) Sum of Edges (w)

Figure 2.13. Summing over all pairs of abstract states to get the weight function w.

where $P_L(i)$ is the probability of state i in the left (first) distribution, and $P_R(j)$ is the probability of state j in the right (second) distribution.

If there is a flow between s and t with weight 1.0, then the matching for the abstract state pair $\bar{s}_l$, $\bar{s}_r$ succeeds. If every pair of abtsract states succeeds, the two distributions are compatible, and the weight function w (whether it is w_λ or w_{ija}) is defined by the weight of the flow over the edges. For any pair $i, j \in S$:

$$w(i,j) = flow(l_i, r_j) \cdot \sum_{i' \in [i]_f} P_L(i') \cdot \sum_{j' \in [j]_{f_\eta}} P_R(j')$$

where the flow for i, j is the flow taken the graph for the left and right abstract states (see Figure 2.13).

This weight function obeys Equations 2.31 - 2.33, in the case of w_{ija}, or Equations 2.34 - 2.36, in the case of w_λ. Equations 2.31 and 2.34 follow from the fact that there is no available capacity between incompatible left and right vertices, therefore $w(i,j)$ is 0 when i and j are incompatible.

Equations 2.32 - 2.33 or 2.35 - 2.36 can be shown to be true by examining the total in and out flow at each node in L and R.

Lemma 2.12. *When the matching algorithm succeeds for the abstract state pair $\bar{s}_r \in \bar{S}_\eta$ and $\bar{s}_l \in \bar{S}$, for any $i \in \bar{s}_l$:*

$$\sum_{j \in \bar{s}_r} w(i,j) = P_L(i) \cdot \sum_{j \in \bar{s}_r} P_R(j)$$

Proof. When the matching algorithm succeeds for the pair $\bar{s}_r, \bar{s}_l$, the flow out of the source node is 1. However, the total available capacity of the edges leaving the source is also 1:

$$\begin{aligned}\sum_{l_i \in L} cap(s, l_i) &= \sum_{l_i \in L} \frac{P_L(i)}{\sum_{i' \in \bar{s}_l} P_L(i')} \\ &= \frac{P_L(\bar{s}_l)}{P_L(\bar{s}_l)} \\ &= 1.\end{aligned}$$

This means that the full capacity of each edge leaving the source node must be used to achieve a flow of 1.0. This in turn implies that the total incoming or outgoing flow through any left node l_i where $i \in \bar{s}_l$ is equivalent to the full capacity from the source to that node, $cap(s, l_i)$:

$$\sum_{j \in \bar{s}_r} flow(l_i, r_j) = cap(s, l_i) \qquad \text{total in/out flow at node } l_i$$

Thus, for any $\bar{s}_r \in \bar{S}_\eta$:

$$\begin{aligned}\sum_{j \in \bar{s}_r} w(i,j) &= \sum_{j \in \bar{s}_r} flow(l_i, r_j) \cdot \left[\sum_{i' \in [i]_f} P_L(i') \cdot \sum_{j' \in [j]_{f_\eta}} P_R(j') \right] \\ &= \sum_{j \in \bar{s}_r} flow(l_i, r_j) \cdot \sum_{i' \in [i]_f} P_L(i') \cdot \sum_{j' \in \bar{s}_r} P_R(j') && f_\eta(j) = \bar{s}_r \\ &= cap(s, l_i) \cdot \sum_{i' \in [i]_f} P_L(i') \cdot \sum_{j' \in \bar{s}_r} P_R(j') && \text{replace in flow with out flow} \\ &= \frac{P_L(i)}{\sum_{i' \in [i]_f} P_L(i')} \cdot \sum_{i' \in [i]_f} P_L(i') \cdot \sum_{j' \in \bar{s}_r} P_R(j') && \text{Equation 2.51} \\ &= P_L(i) \cdot \sum_{j' \in \bar{s}_r} P_R(j') && \text{Cancel terms}\end{aligned}$$

□

A similar proof yields the analogous lemma for the right hand distribution.

Lemma 2.13. *When the matching algorithm succeeds for $\bar{s}_r \in \bar{S}_\eta$ and $\bar{s}_l \in \bar{S}$, for any $j \in \bar{s}_r$:*

$$\sum_{i \in \bar{s}_l} w(i,j) = \sum_{i \in \bar{s}_l} P_L(i) \cdot P_R(j)$$

Proof. The flow at any right side node r_j is $cap(r_j, t)$ when the algorithm succeeds, so that:

$$\sum_{j \in \bar{s}_l} flow(l_i, r_j) = cap(r_j, t) \qquad \text{total in/out flow at node } r_j.$$

Therefore, for any $\bar{s}_l \in \bar{S}$

$$\begin{aligned}
\sum_{i \in \bar{s}_l} w(i,j) &= \sum_{i \in \bar{s}_l} flow(l_i, r_j) \cdot \sum_{i' \in [i]_f} P_L(i') \cdot \sum_{j' \in [j]_{f_\eta}} P_R(j') \\
&= \sum_{i \in \bar{s}_l} flow(l_i, r_j) \cdot \sum_{i' \in \bar{s}_l} P_L(i') \cdot \sum_{j' \in [j]_{f_\eta}} P_R(j') && f(i) = \bar{s}_l \\
&= cap(r_j, t) \cdot \sum_{i' \in \bar{s}_l} P_L(i') \cdot \sum_{j' \in [j]_{f_\eta}} P_R(j') && \text{total out flow at node } r_j \\
&= \frac{P_R(j)}{\sum_{j' \in [j]_{f_\eta}} P_R(j')} \cdot \sum_{i' \in \bar{s}_l} P_L(i') \cdot \sum_{j' \in [j]_{f_\eta}} P_R(j') \\
&= P_R(j) \cdot \sum_{i' \in \bar{s}_l} P_L(i')
\end{aligned}$$

□

These two lemmas can be used to show that Equations 2.32, 2.33, 2.35 and 2.36 are satisfied. All four proofs all follow the same basic outline, shown below for Equation 2.32.

Lemma 2.14. *If w_{ija} is a weight matching found by Procedure 2.5.3 for the next state distributions $P(S \mid i, a)$ and $P(S \mid j, a)$, where $i, j \in S$ and $a \in A$ then Equation 2.35 is satisfied. That is, for any abstract state $\bar{s}_r \in \bar{S}_\eta$:*

$$\sum_{l \in \bar{s}_r} w_{ija}(k,l) = P(k \mid i, a) \cdot P(\bar{s}_r \mid j, a).$$

Proof. If P_L is $P(S \mid i, a)$, and P_R is $P(S \mid j, a)$:

$$\begin{aligned}\sum_{l \in \bar{s}_r} w_{ija}(k,l) &= P_L(k) \cdot \sum_{l' \in \bar{s}_r} P_R(l') \\ &= P(k \mid i, a) \cdot \sum_{l' \in \bar{s}_r} P(l' \mid j, a) \\ &= P(k \mid i, a) \cdot P(\bar{s}_r \mid j, a).\end{aligned}$$

□

Procedure 2.5.1 determines whether or not Equations 2.34 - 2.36 can be satisfied by $\bar{M}_\chi$ and $\bar{M}_\eta$.

2.5.3 Time Analysis

The run time needed to compute the compatibility function dominates the computational complexity of Procedure 2.5.1. In the worst case:

- All state pairs are incompatible, and one pair is marked as incompatible in each iteration of the outer "repeat" loop of Procedure 2.5.2. The outer loop then executes $O(|S|^2)$ times.
- The inner "for" loop of Procedure 2.5.2 has $O(|S|^2 \cdot |A|)$ iterations.

Within the inner for loop, two distributions over next states are checked for compatibility. In this section, this check was described in terms of a set of small graph flow problems. However, these can be transformed into a multi-source multi-sink Maximum Flow problem with $O(|S|)$ vertices and up to $O(|S|^2)$ edges (in the worst case, when all states are compatible). There are many different ways of finding solving the maximum flow problem. In the experiments for this work, the Edmons-Karp algorithm was used (Cormen et al., 2009), with running time $O(V \cdot E^2)$, where V is the number of vertices and E the number of edges. This implies that each flow graph takes $O(|S|^5)$ in the worst case.

The total running time is therefore polynomial, though the exponent is quite high:

$$O(|S|^2 \cdot |S|^2 \cdot |A| \cdot |S|^5) = O(|A| \cdot |S|^9)$$

In practice, the running time of the graph flow algorithm decreases as more states are found to be incompatible, so that all of the worst case assumptions are unlikely to be true at the same time on any single iteration of the outer repeat loop. Nonetheless, while this algorithm is technically polynomial time, it has a very high exponent.

2.6 Comparison of Shadow Model and Compatibility Tests

Domain	Observation Map (κ)	Shadow Proc 2.4.1	Compat. Proc 2.5.1	Sim
Figure 2.3	$\{c \wedge l, c \wedge g, c \wedge b\}$, $\{\neg c \wedge l, \neg c \wedge g, \neg c \wedge b\}$	pass	pass	pass
Figure 2.4	$\{c \wedge l, c \wedge g, c \wedge b\}$, $\{\neg c \wedge l, \neg c \wedge g, \neg c \wedge b\}$	fail	fail	fail
Figure 2.4	$\{c \wedge l\}$, $\{c \wedge g\}$, $\{c \wedge b\}$, $\{\neg c \wedge l\}$, $\{\neg c \wedge g\}$, $\{\neg c \wedge b\}$	pass	pass	pass
Three halls (Figure 2.8)	$\{w\}$, $\{lg, g, b\}$, $\{c\}$, $\{a\}$	fail	fail	fail
	$\{w\}$, $\{lg, b\}$, $\{g\}$, $\{c\}$, $\{a\}$	fail	pass	pass
	$\{w\}$, $\{lg\}$, $\{b\}$, $\{g\}$, $\{c\}$, $\{a\}$	fail	pass	pass

Table 2.1. Comparison of Procedures 2.4.1 and 2.5.1, and a direct simulation of 10,000 belief states ("Sim" column). In the observation map column, observations are identified by their first letter, except in the case of $lightgrey$ (lg) and cat (a). Each set of observation symbols represents a single abstract observation.

Table 2.1 compares the shadow model and compatibility function tests when applied to the POMDPs defined thus far (Figures 2.3, 2.4 and 2.8). In each case, the outcomes of the two tests are compared to a direct verification of the accuracy of the abstract model. The "Sim" column reports the result of $10,000$ steps of random exploration in the specified POMDP. At each step, both b_h and $\bar{b}_{\chi(h)}$ were calculated. The abstract state mapping function f was then applied to b_h, and the results compared to $\bar{b}_{\chi(h)}$. If at every step the results matched, the abstract model was reported as passing the simulation test, otherwise

the abstract model was reported to be inadequate in simulation. For the size of POMDP in each of these examples, this should provide a reasonable approximation of a direct test of Equation 2.12 for all h, although only $10,000$ histories were tested.

2.7 Improving the Observation Map

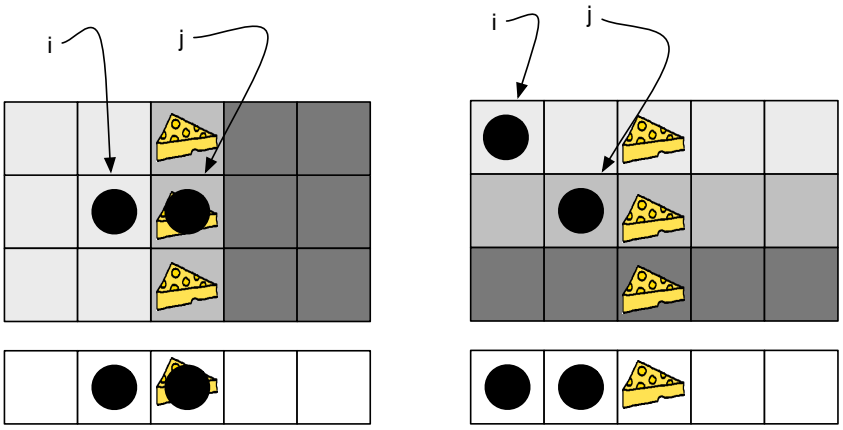
With an evaluation procedure like Procedure 2.4.1 or 2.5.1, it may be possible in some cases to exhaustively search and test all possible candidate observation mapping functions. However, this approach can be expensive, and directed search methods for finding candidates are generally preferred. Procedure 2.3.1 is the outline of a directed search method. It begins with the observation function κ induced by ζ. The remainder of the algorithm is a loop that iteratively improves the observation map. This section defines the iterative improvement step of the search algorithm.

None of the algorithms in this section are guaranteed to find the most compact observation map possible. Each algorithm presented here uses the information about how the observation mapping function κ fails the tests outlined in Section 2.4 and Section 2.5 to determine what portions of κ must change, however, they all use heuristics to determine how to enforce those changes by updating κ.

Procedure 2.7.1, as one example, is the simplest algorithm for observation improvement in this Section. It refines the abstract observation clusters whenever doing so would help to distinguish between any two abstract states. Take, for example, the domain in Figure 2.4(a), when the abstract observation function is:

- Abstract observation 1: all observations with $cheese$
- Abstract observation 2: all observations with $\neg cheese$.

with abstract model shown in Figure 2.4(b). The fact that $lightgrey$ is observed only in the two leftmost abstract states, while $grey$ is only observed in the middle abstract state indicates that these two observations distinguish between these groups of states. In other



(a) Vertical Stripes (b) Horizontal Stripes

Figure 2.14. Two hypothetical belief states for which the *lightgrey*/*grey* feature distinction would be useful.

Procedure 2.7.1 simplerImproveObservationMapAlgorithm(M, f, κ)

```
for all i, j ∈ S do
  if f(i) ≠ f(j) then
    κ ⇐ distinguishBetweenStates(i, j, κ)
return κ
```

words, if the agent had a belief distribution like the one shown in Figure 2.14(a), observing $lightgrey$ or $grey$ would refine the belief distribution, and would improve the agent's estimate of its abstract state. Procedure 2.7.1 outlines a method that implements this principle in the simplest manner possible.

Note that Procedure 2.7.1 does not reference the shadow model, or the state compatibility function. It examines each pair of states i, j in S, and distinguishes between each pair of states with different abstract labels. Procedure 2.7.5 will address in more detail exactly how the method $distinguishBetweenStates$ refines the observation map, as this will be an important step of the final observation splitting algorithm. However, there is a problem with the way that Procedure 2.7.1 chooses pairs of states to examine.

The simplicity of Procedure 2.7.1 method is appealing. However, it fails on some fairly simple test cases, such as Figure 2.3. Take, for example, the two states i and j in Figure 2.14(b). These states

(a) have different abstract labels in the abstract model shown

(b) have different observation distributions

This seems to indicate that the observation distinction that helps to distinguish between these two states, namely the $lightgrey/grey$ feature distinction, is important, and it would be — if all belief states $b : S \rightarrow [0, 1]$ were reachable from the initial belief distribution. However, in general, this assumption is not true: B_M (the set of reachable belief states, Equation 2.5) is often a smaller subset of the set of valid belief distributions. B_M is constrained by the initial belief state, and the transition structure of the POMDP. It is this fact that the shadow model and compatibility function approaches exploit. Given the specific b_λ and transition function described for Figure 2.3, the belief distribution shown in Figure 2.14(b) can never occur, and in fact there is no belief state in B_M for which the distinction between $lightgrey$ and $grey$ improves the abstract belief estimate. This is not to say that no belief state places probability mass on both i and j. The initial belief state (see Figure 2.6(a)) is one such example.

What is required is a more precise method of searching for pairs of states where the distinction between the two states has an effect on the abstract projection of some belief state in B_M. It is these pairs of states that should be distinguished from one another on the basis of their observation functions. The compatibility function provides one tool for identifying these states.

The observation mapping function evaluation methods in Procedures 2.4.1 and 2.5.1 construct data structures (the shadow model and compatibility function) that can be used to pinpoint pairs of states like this. The algorithm for observation map improvement has two essential components:

Procedure 2.7.2 improveObservationMap(M, κ)

$compatibleStates \Leftarrow$ constructCompatibilityFunction(M, κ, f_η, f) // Procedure 2.5.2
$Q \Leftarrow$ an empty set of state pairs (implemented as a queue)
makeDistributionsCompatible($b_\lambda, b_\lambda, compatibleStates, Q$) // Procedure 2.7.3
while Q not empty **do**
  $\langle i, j \rangle \Leftarrow$ an element removed from Q
  $\kappa \Leftarrow$ makeObservationsCompatible(i, j, κ) // Procedure 2.7.5
  for all $a \in A$ **do**
    // This step may add state pairs to Q
    makeDistributionsCompatible($P(S \mid i, a), P(S \mid j, a), compatibleStates, Q$) // Procedure 2.7.3
    // mark the two states as compatible
    $compatibleStates(i, j) \Leftarrow$ **true**
return κ

- Searching for pairs of states that should be made compatible (Procedures 2.7.2 and 2.7.3).

- Splitting observations so that states become compatible (Procedure 2.7.5).

Both of these steps are part of a recursive process that forces pairs of states to become compatible, and in the process refines the observation map.

2.7.1 Merging Distributions

If Procedure 2.5.1 fails and returns $false$, it must be the case that Procedure 2.5.3 failed for the initial belief distribution. Therefore, repair of an observation map begins at the initial belief distribution (see Procedure 2.7.2). Failure can depend on observations in states that are several steps removed from the initial belief distribution (see Figure 2.8). Procedure 2.7.3 locates these dependencies by iterating through state pairs, working forward from the initial belief distribution b_λ.

There must be some set of pairs of states $Q \subset S \times S$ that would make b_λ compatible with itself. Procedure 2.7.3 creates one such set Q. It starts with the best possible graph flow found using the existing state compatibility function ($flow$). Next, Procedure 2.7.3 adds pairs of states to Q, using three steps:

Procedure 2.7.3 makeDistributionsCompatible($P_L(S), P_R(S), compatibleStates, Q$)

for all $\bar{s}_l, \bar{s}_r \in \bar{S}$ **do**
  $graph \Leftarrow$constructDistributionGraph($P_L(S), \bar{s}_l, P_R(S), \bar{s}_r, compatibleStates$)
  $flow \Leftarrow$ maxFlow($graph$)
  if $\sum_i flow(s, l_i) < 1$ **then**
    // Add edges as necessary between incompatible states
    $augmentedGraph \Leftarrow$ augmentedMatching($P_L(S), \bar{s}_l, P_R(S), \bar{s}_r$)
    $newFlow \Leftarrow$ maxFlow($initalFlow = flow, augmentedGraph$)

    // If an edge was added between two states, make the paired states compatible
    for all edges (l_i, r_j) where newFlow(l_i, r_j) $>$ flow(l_i, r_j) **do**
      // add the pair i, j to the merge queue
      $Q \Leftarrow Q \cup \langle i, j \rangle$

- Assume all states are compatible, and add edges of capacity 1 between all states in the left and right distributions of the matching graph accordingly.

- Initialize the flow in this augmented graph using $flow$. This ensures that existing edges between compatible state pair edges are used before edges between other pairs of states.

- Calculate a maximum capacity flow from the source to sink.

- For each new edge in the augmented flow that was not in $flow$, add the two states corresponding to the nodes it connects to Q

The method assumes that any pair of states could become compatible. In this implementation, there is no preference given to which pairs of states are forced to become compatible. This means that the matching found is somewhat arbitrary. This is one of the reasons that this method cannot be guaranteed to find the optimal κ refinement.

2.7.2 Observation Splits

There are numerous supervised learning methods that search for features with discriminative power, and in practice implementing some variation on one of these algorithms is probably the best solution for implementing Procedure 2.7.5. However, in keeping with the

Procedure 2.7.4 augmentedMatchingGraph($P_L(S), \bar{s}_l, P_R(S), \bar{s}_r$)

$V \Leftarrow \{s, t\}$ // new set of vertices
$capacity : V \times V \rightarrow I\!R$ // capacity of the edges

// Fill in the nodes and edges of the graph
for all $i \in \bar{s}_l$ **do**
  $V \Leftarrow V \cup \{l_i\}$
  $capacity(s, l_i) = \frac{P_L(i)}{P_L(f(i)}$
for all $j \in \bar{s}_r$ **do**
  $V \Leftarrow V \cup \{r_j\}$
  $capacity(r_j, t) = \frac{P_R(j)}{P_R(f_\eta(j))}$
for all $i \in \bar{s}_l$**,** $j \in \bar{s}_r$ **do**
  $capacity(l_i, r_j) = 1$

return $graph = (V, capacity)$

Procedure 2.7.5 distinguishBetweenStates(i, j, κ)

// Construct a new observation mapping function such that:
$\kappa(o_n) = \kappa(o_m) \rightarrow \frac{P(o_n|i,a)}{P(o_m|i,a)} = \frac{P(o_n|j,a)}{P(o_m|j,a)}$

goal of treating each observation as a discrete entity, in this section we'll discuss the exact solution to the problem of finding discriminative observation splits. The main purpose of this exercise is to clarify the difficulty of finding an exact solution, and to categorize the problem, in hopes of finding good approximation algorithms. We will consider two scenarios:

1. Equation 2.13 is satisfied, so that for all states s actions a and observations o, $P(o \mid s, a) > 0$.

2. Equation 2.13 is not satisfied.

The first scenario is straightforward, while the second actually results in a NP-complete problem. This is not surprising, given the difficulty of learning problems in general.

For two states to become compatible, their observation functions must be made compatible, in Procedure 2.7.5. This procedure splits the observations into new groups such that:

$$\kappa(o_n) = \kappa(o_m) \rightarrow \frac{P(o_n|i,a)}{P(o_m|i,a)} = \frac{P(o_n|j,a)}{P(o_m|j,a)} \tag{2.54}$$

for the two states i and j in S, and all actions a.

This procedure is simplest when $P(o \mid s,a) > 0$ for all states, actions and observations (as in Section 2.4). In this case, Equation 2.54 induces an equivalence relation over observations. Two observations are equivalent, and can thus share the same abstract label, exactly when the ratios for the two observations are equivalent for the two states i and j in S. Rewriting Equation 2.54 to make the equivalence relation a bit clearer yields:

$$\kappa(o_n) = \kappa(o_m) \iff \forall i,j,a, \frac{P(o_n|i,a)}{P(o_n|j,a)} = \frac{P(o_m|i,a)}{P(o_m|j,a)}$$

The ratio $\frac{P(o|i,a)}{P(o|j,a)}$ is a real valued key for the observation o. All observations with the same key are equivalent, and clustered under the same abstract label.

Lemma 2.15. *Equation 2.54 implies that states i and j have the same shadow observation functions (Equation 2.17).*

Proof. We must show that Equation 2.54 implies that for all observations $o \in O$, $\frac{P(o|i,a)}{P(\kappa(o)|i,a)} = \frac{P(o|j,a)}{P(\kappa(o)|j,a)}$, indicating that as far as the observation function is concerned, i and j are observation compatible ($i \sim_o j$, Equation 2.25).

$$
\begin{aligned}
\frac{P(o \mid i,a)}{P(\kappa(o) \mid i,a)} &= \frac{P(o \mid i,a)}{\sum_{o_m \in [o]_\kappa} P(o_m \mid i,a)} \\
&= \frac{1}{\frac{\sum_{o_m \in [o]_\kappa} P(o_m|i,a)}{P(o|i,a)}} && \text{Both num. and denom. are not } 0 \\
&= \frac{1}{\sum_{o_m \in [o]_\kappa} \frac{P(o_m|i,a)}{P(o|i,a)}} \\
&= \frac{1}{\sum_{o_m \in [o]_\kappa} \frac{P(o_m|j,a)}{P(o|j,a)}} && \text{by Equation 2.54} \\
&= \frac{P(o \mid j,a)}{\sum_{o_m \in [o]_\kappa} P(o_m \mid j,a)} && \text{Both num. and denom. are not } 0 \\
&= \frac{P(o \mid j,a)}{P(\kappa(o) \mid j,a)}
\end{aligned}
$$

□

The problem becomes more complicated when observation probabilities can equal 0. Any observation which is never observed from either state i or state j is not useful for distinguishing between the two states, and should not be affected by the observation splits incurred when distinguishing between i and j. Let O_{ija} be the set of observations that could be observed in either state i or state j after action a:

$$O_{ija} = \{o \in O \mid P(o \mid i,a) > 0 \vee P(o \mid j,a) > 0)\}. \tag{2.55}$$

Any observation map κ which satisfies Equation 2.54 for the observations in O_{ija} is sufficient. The abstract labels of observations not in O_{ija} are not specified (thus this is obviously not an equivalence relation).

Lemma 2.16. *If Equation 2.54 holds for all observations $o \in O_{ija}$, then states i and j have compatible observation functions (Equation 2.25).*

Proof. We must show that if Equation 2.54 is true for all observation pairs in O_{ija}, then whenever $P(\kappa(o) \mid i,a)$ is greater than 0 and $P(\kappa(o) \mid j,a)$ is greater than 0, it follows that $\frac{P(o|i,a)}{P(\kappa(o)|i,a)} = \frac{P(o|j,a)}{P(\kappa(o)|j,a)}$ (Equation 2.25).

There are three cases: the observation o can have zero probability in both state i and state j, in either one of the states, or in neither state.

1. Case 1: $P(o \mid i, a) = 0$ and $P(o \mid j, a) = 0$. In this case, o is not in $O_{ija}$and Equation 2.54 does not apply. Since the probability of o is zero in both states, the equality is trivially true:

$$\begin{aligned} \frac{P(o \mid i, a)}{P(\kappa(o) \mid i, a)} &= 0 \\ &= \frac{P(o \mid j, a)}{P(\kappa(o) \mid j, a)} \end{aligned}$$

2. Case 2: Without loss of generality, assume that $P(o \mid i, a) = 0$ and $P(o \mid j, a) \neq 0$. In this case o is in O_{ija}, and Equation 2.54 applies.

   For all $o_m \in [o]_\kappa$:

$$\begin{aligned} \frac{P(o_m \mid i, a)}{P(o_m \mid j, a)} &= \frac{P(o \mid i, a)}{P(o \mid j, a)} && \text{by Equation 2.54} \\ &= 0. \end{aligned}$$

   This can only be true if $P(o_m \mid i, a) = 0$. Since this is true for all $o_m \in [o]_\kappa$:

$$\begin{aligned} P(\kappa(o) \mid i, a) &= \sum_{o_m \in [o]_\kappa} P(o_m \mid i, a) \\ &= 0. \end{aligned}$$

   Therefore, $P(\kappa(o) \mid i, a) = 0$ and the implication in Equation 2.25 is true due to the fact that the preconditions are false.

3. Case 3: $P(o \mid i, a) \neq 0$ and $P(o \mid j, a) \neq 0$. In this case o is in O_{ija}, and Equation 2.54 applies.

   This case follows the same proof as Lemma 2.15.

Procedure 2.7.6 observationCompatibilityFunction($i, j, \kappa, \sim_\kappa$)

$\sim_\kappa$: $O \times O \rightarrow \{$**true**, **false**$\}$ // Initiailize all to **true**
for all $a \in A$ **do**
  for all $o_k, o_l \in O$ **do**
    if $[P(o_k \mid s_i, a) > 0] \vee [P(o_k \mid s_j, a) > 0]$ **then**
      if $[P(o_l \mid s_i, a) > 0] \vee [P(o_l \mid s_j, a) > 0]$ **then**
        if $\left[\frac{P(o_k|s_i,a)}{P(o_k|s_j,a)} \neq \frac{P(o_l|s_i,a)}{P(o_l|s_j,a)}\right]$ **then**
          $o_k \sim_\kappa o_l \Leftarrow$ **false**

□

Procedure 2.7.6 constructs an observation compatibility function $\sim_\kappa$: $O \times O \rightarrow \{true, false\}$ that satisfies the preconditions for Lemma 2.16. Any observation mapping function κ which clusters the observations into compatible groups, so that $\kappa(o_i) = \kappa(o_j) \rightarrow o_i \sim_\kappa o_j$ will provide the desired changes in the state compatibility function, required by Procedure 2.7.2.

Consider the domain from Figure 2.8, with output function:

$$\zeta(\{cheese\}) \stackrel{\text{def}}{=} +5$$
$$\zeta(\{cat\}) \stackrel{\text{def}}{=} -10$$
$$\zeta(\{lightgrey\}) \stackrel{\text{def}}{=} 0$$
$$\zeta(\{grey\}) \stackrel{\text{def}}{=} 0$$
$$\zeta(\{black\}) \stackrel{\text{def}}{=} 0$$
$$\zeta(\{white\}) \stackrel{\text{def}}{=} 0$$

Applying Procedure 2.7.2 implemented using with Procedure 2.7.5 to this domain (as though it satisfied Equation 2.13) yields the identity observation mapping function, with no grouped obsevations[3].

[3]Undefined ratios were treated as $\perp$ in this experiment.

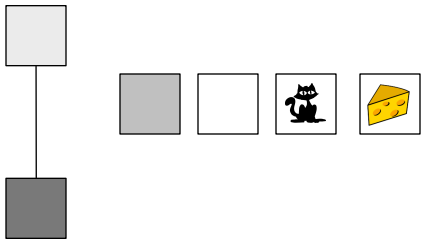


Figure 2.15. Observation compatibility graph for the POMDP of Figure 2.8. Compatible observations are linked by edges.

However, the following ζ respecting mapping function passes the compatibility test (Procedure 2.5.1), as shown in Table 2.1:

$$\bar{o}_1 \stackrel{\text{def}}{=} \{cheese\}$$

$$\bar{o}_2 \stackrel{\text{def}}{=} \{cat\}$$

$$\bar{o}_3 \stackrel{\text{def}}{=} \{grey\}$$

$$\bar{o}_4 \stackrel{\text{def}}{=} \{black, lightgrey\}$$

$$\bar{o}_5 \stackrel{\text{def}}{=} \{white\}$$

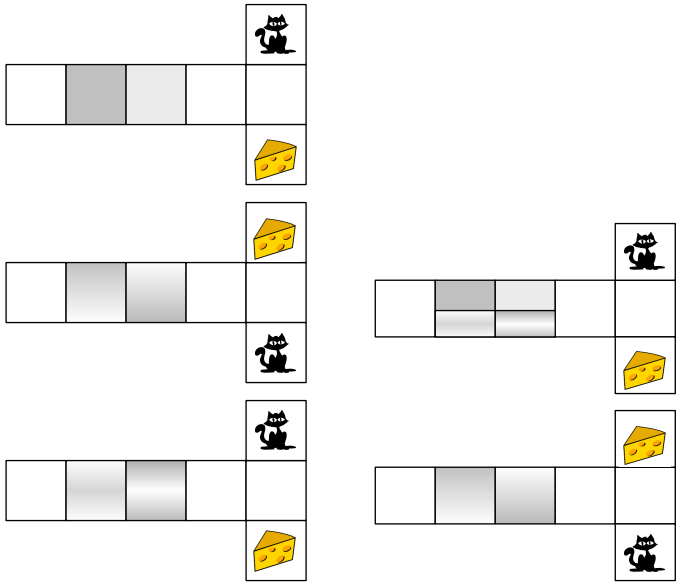
and yields a smaller abstract state set than the original observation set. This mapping function would satisfy the prerequisites for Lemma 2.16. The observation splitting algorithm starts with the following observation map (split according to ζ):

$$\bar{o}_1 \stackrel{\text{def}}{=} \{cheese\}$$

$$\bar{o}_2 \stackrel{\text{def}}{=} \{cat\}$$

$$\bar{o}_3 \stackrel{\text{def}}{=} \{grey, black, lightgrey, white\}$$

Given this initial observation split, and its associated state compatibility function, Procedure 2.7.6 can be applied to find an observation compatibility function. This observation



(a) POMDP (b) Abstract Model

Figure 2.16. Three corridor gridworld POMDP with two noisy observation markers in each hallway. In this case there are two color markings in each hallway, which signal the type of hallway the agent is in, as well as the location within that hallway. However, the observation of these markers is noisy (see POMDP definition in text for details).

compatibility function is shown in Figure 2.15. Observations $lightgrey$ and $black$ (the only two compatible observations) can remain clustered into one abstract observation, resulting in the correct observation mapping function.

In this particular example, it is clear which observations should be clustered, however, in general this is not the case. Take the case of the POMDP illustrated in Figure 2.16, defined as follows:

States: Each square in Figure 2.16 represents a state.

Actions: up, $down$, $left$, $right$

Transitions: Actions fail with a small probability ϵ. Failure results in no change to the state.

Observations: $white$, $lightgrey$, $grey$, top, $bottom$, $middle$, $sides$, $cheese$, cat

Observation Function: The obserations $white$, $cheese$ and cat are deterministically observed. In the marked states in each corridor, the signaling observations may be confused with one another:

- $grey$ may be mistaken for $lightgrey$ with probability ϵ, and vice versa.
- top may be mistaken for $bottom$ with probability ϵ, and vice versa.
- $middle$ may be mistaken for $sides$ with probability ϵ, and vice versa.

Initial Belief State: Equal probability mass on the leftmost state in each corridor.

Given the initial obsevation split:

$$\bar{o}_1 \stackrel{\text{def}}{=} \{cheese\}$$
$$\bar{o}_2 \stackrel{\text{def}}{=} \{cat\}$$
$$\bar{o}_3 \stackrel{\text{def}}{=} \{grey, lightgrey, top, bottom, middle, sides, white\}$$

Procedure 2.7.6 yields the observation compatibility relation shown in Figure 2.17. There are at least two ways of clustering the observations to respect this compatibility relation:

- $\{white\}, \{top\}, \{bottom\}, \{middle, grey\}, \{sides, lightgrey\}, \{cheese\}, \{cat\}$
- $\{white\}, \{top\}, \{bottom\}, \{middle, lightgrey\}, \{sides, grey\}, \{cheese\}, \{cat\}$.

While both observation groupings obey the observation compatibility constraints, only the first one results in the abstract model shown in Figure 2.16(b). The other choice results in a larger abstract state set, in which all three corridors separate. This is easy to detect in this case: there are only two choices to test, and the first option allows states which had the same abstract state label before Procedure 2.7.5 was called to keep the same abstract label, while the second option does not.

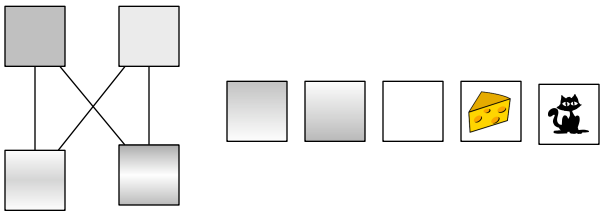


Figure 2.17. Observation compatibility graph for the POMDP of Figure 2.16. Compatible observations are linked by edges.

The basic problem is determining a compatible observation mapping function κ that wherever possible allows abstract states to remain intact, rather than splitting them. In this particular case, a simple heuristic can be applied to mend the abstract states. However, in general the following question is computationally complex, even for a single pair of states:

> Given two states i and j, where at the last iteration $f(i) = f(j)$, and an observation compatibility function $\sim_\kappa$, is there an observation mapping function κ that satisfies Lemma 2.16 for $\sim_\kappa$ and allows i and j to remain part of the same abstract state?

Answering this question is NP-hard. This can be shown by reducing the NP-hard decision version of the Knapsack problem Cormen et al. (2009) to this question. The decision version of the Knapsack Problem is as follows: given a knapsack of fixed size k, and various objects of weight $W = \{w_0, w_1 ... w_n\}$, determine if there is a set U, $U \subseteq W$, such that $\sum_{w_i \in U} w_i = k$.

Theorem 2.17. *The Knapsack Problem can be reduced to the problem of determining whether, given a particular set of observation constraints, the observation mapping function can be constructed such that two states i and j can have the same abstract label ($f(i) = f(j)$).*

Proof. Construct a POMDP M with states i and j such that i and j can only have the same abstract label if it is possible to exactly fill the knapsack.

The POMDP must have two states: s_0 corresponds to the items to be placed in the knapsack, and s_1 corresponds to the knapsack. The observation function for these states under some specific action a will be as follows:

- s_2: one observation o_i corresponding to each $w_i \in W$:
  - $P(o_i \mid s_0, a) = \frac{w_i}{t}$

  where $t = \sum_{w_i \in W} w_i$.
- s_1 emits two observations:
  - o_k: $P(o_k \mid s_1, a) = k/t$ (knapsack)
  - o_x: $P(o_x \mid s_1, a) = 1 - P(o_k \mid s_1, a)$ (all remaining probability).

Both s_0 and s_1 should have the same next state distribution, so that for all states s', $P(s' \mid s_0, a) = P(s' \mid s_1, a)$. This means that the only distinction between the two states lies in their observation functions. If the abstract observation functions for the two states is the same, it will be the case that $f(s_0) = f(s_1)$.

Assume that the other states and observations of the POMDP, and the output function ζ can be constructed so that the observation compatibility function for the observations of s_0 and s_1 is:

$$o_k \nsim_\kappa o_x$$

and for all o_i corresponding to $w_i \in W$:

$$o_i \sim_\kappa o_k$$

$$o_i \sim_\kappa o_x.$$

For all pairs o_i, o_j corresponding to pairs $w_i, w_j \in W$:

$$o_i \sim_\kappa o_j.$$

Since $o_k \nsim_\kappa o_x$, s_1 must emit two abstract observations, $\bar{o}_0 = \kappa(o_k)$ and $\bar{o}_1 = \kappa(o_x)$. If $f(s_0) = f(s_1)$, then it must be the case that for all $\bar{o} \in \bar{O}$:

$$P(\bar{o} \mid s_0, a) = P(\bar{o} \mid s_1, a)$$

therefore:

$$P(\bar{o}_0 \mid s_0, a) = P(\bar{o}_0 \mid s_1, a)$$
$$P(o_k \mid s_0, a) = \sum_{o_i \in \bar{o}_0} P(o_i \mid s_1, a)$$

Let U be the set $\{w_i \in W \mid \kappa(o_i) = \bar{o}_0\}$. Then:

$$k/t = \sum_{w_i \in U} w_i/t$$
$$k = \sum_{w_i \in U} w_i$$

and U is the set of weights needed to fill the knapsack. This indicates that whenever κ can be found such that i and j have the same abstract observation function, the corresponding knapsack problem has a solution.

Similarly, if the knapsack problem has a solution U, then the observation mapping function should be constructed such that for all o_i where w_i is in U, $\kappa(o_i) = \bar{o}_0$. All other observation emitted from s_0 should have abstract label $\kappa(o_i) = \bar{o}_1$. With this observation mapping function definition, $f(s_0) = f(s_1)$.

Thus, the knapsack problem has a solution if and only if κ can be constructed such that $f(s_0) = f(s_1)$. □

2.8 Time Complexity

Procedure 2.7.2 has several parts.

- The outer repeat loop may iterate for up to $|O|$ steps, as the abstract observation map must change on each iteration.
- The time to construct the compatibility function. This is dominated by the time needed to merge state pairs, in the worst case.

The remainder of each iteration is devoted to merging state pairs.

- At most $|S|^2$ state pairs may be merged (all state pairs), with $|A|$ actions to be examined per state pair.
- Each observation distribution merge requires at least $|O|^2 \cdot |A|$ steps to construct $\sim_\kappa$. For the purposes of this analysis, the heuristic that chooses among the possible observation maps will be assume to require time $O(n)$.
- Each next state distribution merge requires two steps of graph flow construction. Each step is $O(|S|^5)$, as shown in Section 2.5.3.

The overall complexity of the algorithm is therefore:

$$O\left(|O| \cdot |S|^2 \cdot |A| \cdot \left[\left(|O|^2 \cdot |A| + n\right) + |S|^5\right]\right).$$

Depending on the ratio between the number of states, actions and observations, different elements of this formula will dominate the running time. The most variable portion of this term is the number of state pairs to be merged.

2.9 Conclusion

This chapter defined two acceptance criteria for output-directed abstract POMDP models. Each of these criteria can be evaluated in polynomial time, but both may reject some

valid abstract models. The first acceptance criteria, the Shadow Model test, works under the assumption that Equation 2.13 is satisfied. It is somewhat faster than the second acceptance criteria, the Shadow Compatibility test. The Shadow Compatibility test accepts a wider range of abstract models, particularly when the Shadow Model assumptions do not hold.

This chapter also defined an algorithm that searches for the smallest abstract models that satisfy these acceptance criteria Section 2.7 specified such an algorithm, although the search strategy is not guaranteed to find the smallest possible satisfying abstraction. The algorithm includes two steps implemented using heuristics: the step that chooses one of several possible sets of state pairs to merge, and the step that chooses one of several improved observation mapping functions. Both of these steps could be improved using better heuristics.

The next chapter will define observation map testing algorithms using Predictive State Representations (PSRs), and examine the reasons that the PSR abstraction approach expands the set of accepted observation maps, improving on the POMDP abstraction approach.

CHAPTER 3

THE KRYLOV BASIS: POMDP TO PSR ABSTRACTION

3.1 Overview

POMDPs are not the only way of modeling partially observable domains. Predictive State Representations (PSRs) (Littman et al., 2001) are an alternative method of modeling partial-observability. Rather than modeling hidden states, PSRs represent state as a set of predictions about future observations. PSRs represent a fully functional alternative to POMDPs.

The last chapter discussed solutions the to he problem of finding an abstract POMDP model from a known POMDP. The known model provided to the algorithm will be termed the "original" model. This chapter examines two questions, in the context of the Shadow Model and Shadow Compatibility tests developed in the previous chapter:

- Can the original POMDP be replaced with a PSR?
- Can the abstract model be constructed as a PSR rather than a POMDP?

The first question is unfortunately not true: at least in their current form, PSRs cannot supply enough information to serve as the original model for these abstraction techniques.

PSRs can, however, serve as the abstract model. This chapter adapts both the Shadow Model and Shadow Compatibility tests and search algorithm for the case when the original model is a POMDP, and the abstract model is a PSR.

Finally, the chapter compares the abstract PSR model acceptance set to the acceptance set for the abstract POMDP. As a result, we define an intermediate type of model, which will be termed an Observation Conditional POMDP.

3.2 Background: Predictive State

Predictive State Representations (PSRs) (Littman et al., 2001) represent partially observable domains through a set of tests and their outcomes. As in a POMDP, a PSR has an action set A and observation set O. A test, much like a history, is a sequence of action observation pairs. If Θ is the set of all possible tests, then the empty test λ is in Θ, and for every test $t \in \Theta$, $\forall a \in A, o \in O, aot \in \Theta$.

Tests can succeed, or fail depending on whether the expected observation sequence is observed. Test success is defined as observation of the specified test observations upon execution of the test actions, test failure is defined as the observation of any other observation sequence.

For a given set of histories H, let $w_H : H \times \Theta \rightarrow I\!R$ be a history specific function mapping each history, test pair (h, t) to the outcome of test t after history h. Test outcomes can be defined in a variety of ways. Existing research has defined two types of PSR: *value-directed* and *observation-directed*. PSR methods differ in how the outcomes of the empty test λ is defined. The outcome of every test aot longer than the empty test λ is defined recursively, in terms of the shorter test t:

$$w_H(h, aot) \quad = \quad P(o \mid ha) \cdot w_H(hao, t),$$

so that all longer test outcomes are defined in terms of the specified λ outcomes.

There are several ways of defining the base case $w_H(h, \lambda)$ for each history. Littman et al. (2001) defines the outcome of the empty test as 1 for every history, so that $w_H(h, \lambda) = 1$ for all h in H. In this case, $w_H(h, t)$ is the probability that the test t would succeed given history h:

$$w_H(h, t) = P(t \mid h)$$

PSRs with this initialization will be termed *observation-directed* PSRs. Poupart and Boutilier (2002) define the outcome of the empty test λ as the expected immediate reward received after history h: $w_H(h, \lambda) = E(r \mid h)$. In this case, $w_H(h, t)$ is the probability that the test t would succeed given history h, multiplied by the expected reward after the sequence h, t:

$$w_H(h, t) = P(t \mid h) \cdot E(r \mid h, t).$$

Poupart and Boutilier (2002) term this a *value-directed* PSR.

Let Q be a set of tests $Q \subseteq \Theta$ which has the following property, for all $h \in H$ and $t \in \Theta$:

$$w_H(h, t) = \sum_{q \in Q} w_Q(q, t) \cdot w_H(h, q)$$

where w_Q is a weight function, $w_Q : Q \times \Theta \rightarrow I\!R$, encoding the weight of each $t \in \Theta$ given $q \in Q$. The set Q is known as the set of "core" tests. PSRs that are represented using this type of linear set of basis tests are known as "linear" PSRs.

A linear PSR can be defined as a tuple $(Q, A, O, \{T_{ao}\}, b_\lambda)$, where:

- Q is the core set of tests,
- A is the set of actions,
- O is the set of observations,
- $\{T_{ao}\}$ is a set of transition matrices, and
- $b_\lambda : Q \rightarrow I\!R$ is the initial belief over Q.

For each action a and observation o, the $Q \times Q$ transition matrix T_{ao} is defined as follows:

$$T_{ao}(i,j) = w_Q(q_i, aoq_j)$$

and the PSR belief update rule can be written:

$$b_{hao} = b_h T_{ao} \tag{3.1}$$

This yields the following belief states:

$$b_h(i) = w_H(\lambda, hq_i)$$

where $q_i \in Q$.

If the PSR is *observation-directed*, then the belief update can be normalized:

$$b_{hao} = \frac{b_h T_{ao}}{b_h e_{ao}} \tag{3.2}$$

where e_{ao} is a vector defined by w_Q:

$$e_{ao}(i) = w_Q(q_i, ao)$$

This yields belief states with the following entries:

$$b_h(i) = w_H(h, q_i)$$

where $q_i \in Q$. Observation probabilities can be derived from these belief states:

$$P(o \mid h, a) = b_h e_{ao}$$

Thus far, the PSR model has been described as a stand-alone model. PSRs are also related to POMDPs, however: any known POMDP can be transformed into either an observation-

directed or value-directed PSR, where the size of Q may be equal to or smaller than the size of the POMDP state set S.

3.2.1 POMDP to PSR Compression

Poupart and Boutilier (2002) and Littman et al. (2001) use the Krylov Basis (Saad, 2003) to construct a core set of tests Q from a POMDP. Given a POMDP $M = (S, A, T, O, \Omega)$ with initial belief b_λ, these algorithms calculate a PSR $\bar{M} = (Q, A, O, \{\bar{T}_{ao}\}, \bar{b}_\lambda)$ that is equivalent to M in its predictions. The resulting PSR may have fewer tests than the POMDP has states, compressing the original POMDP into an equivalent, but more compact model.

Given a POMDP $M = (S, A, T, O, \Omega)$, define POMDP transition matrices T_{ao} for each pair $a \in A$ and $o \in O$, as:

$$T_{ao}(i, j) = P(o \mid s_j, a) \cdot P(s_j \mid s_i, a).$$

Define a set of prediction vectors corresponding to the set of tests: $\{u_t \mid t \in \Theta\}$ such that:

$$u_\lambda(i) = \begin{cases} 1 & \text{in Littman et al. (2001)} \\ r(s_i) & \text{in Poupart and Boutilier (2002)} \end{cases}$$

$$u_{aot} = T_{ao} u_t$$

Each entry $u_t(i)$ corresponds to a prediction about t given state s_i. Combining this vector with the POMDP belief vector b_h:

$$w_H(h, t) = b_h \cdot u_t^T.$$

The core set of tests Q corresponds to a linearly independent subset of the vectors in $\{u_t \mid t \in \Theta\}$

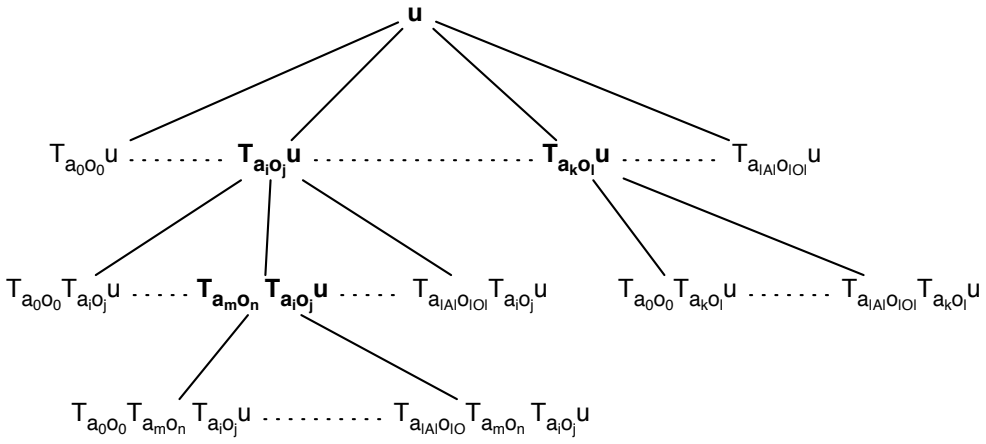


Figure 3.1. The tree of tests for a POMDP. The bolded vectors correspond to the tests λ, $a_i o_j$, $a_k o_l$, and $a_n o_m a_i o_j$ which are chosen to form the core set if tests Q in this hypothetical example. Other tests are not expanded.

Both Littman et al. (2001) and Poupart and Boutilier (2002) use the Krylov Basis $Kr(\{T_{ao}\}, u)$ to calculate Q, where $u = u_\lambda$. The two algorithms differ in their definition of u_λ.

The set of operators $\{T_{ao} \mid a \in A, o \in O\}$ combine to form a branching tree of possible tests, as shown in Figure 3.1. Each node in this tree corresponds to an element of $\{u_t \mid t \in \Theta\}$. The search for a Krylov Basis starts with a single vector u, which forms the root of the test tree shown in Figure 3.1. The full set of tests Θ forms a tree rooted at u. The set Q of core tests form a subtree also rooted at u. The rooted subtree corresponding to Q can be found through breadth first or depth first search of the test tree. If the test at a particular node is not added to Q, the search on that branch terminates, as all children of the node can be eliminated from consideration.

The projection matrix $F : S \times Q \to I\!R$, shown in Figure 3.2, where:

$$F(i, j) = u_{q_j}(s_i)$$

maps POMDP belief vectors over states to PSR belief vectors over tests in Q.

	q_1	q_2	q_3	$\cdots$	q_n
s_0	$u_{q_1}(0)$	$\ldots$			$u_{q_n}(0)$
s_1	$\vdots$	$\ddots$			
s_2					
$\vdots$					
$s_{\lvert S\rvert}$	$u_{q_1}(\lvert S\rvert)$				

Figure 3.2. POMDP Krylov Subspace Projection Matrix

The PSR transition matrices $\bar{T}_{ao}$ and λ weight vector $\bar{u}$ are solutions to the formulas:

$$T_{ao}F = F\bar{T}_{ao} \tag{3.3}$$

$$u = F\bar{u} \tag{3.4}$$

The normalized belief state update rule for a POMDP can be written:

$$b_{hao} = \frac{b_h T_{ao}}{b_h T_{ao} e^T} \tag{3.5}$$

where e is vector of ones. The value-directed PSR does not preserve this update rule. However, the following belief update rule can be used to calculate the un-normalized belief vector:

$$b_{hao} = b_h T_{ao} \tag{3.6}$$

From this point forward, the notation b_h will indicate the belief state as calculated using this update rule, without normalization. The normalized version will be denoted $\mu(b_h)$. Thus we will have:

$$\mu(b_{hao}) = \frac{b_{hao}}{b_{hao} e^T}$$

in the POMDP.

The value-directed PSR algorithms take advantage of the fact that the value function for both the normalized and non-normalized belief states are identical. That is, if v is a vector which maps belief states to values, then the value function for normalized belief states (Equation 3.5) is:

$$\mu(b_h)v = \mu(b_h)r + \gamma \cdot \sum_o (\mu(b_h)T_{ao}e) \cdot \mu(b_{hao})v.$$

If the value function for un-normalized belief vectors is defined as:

$$b_h v = b_h r + \gamma \cdot \sum_o b_{hao} v,$$

then v is the same in the normalized and non-normalized cases (Poupart and Boutilier, 2002):

$$b_h v = b_h r + \gamma \cdot \sum_o b_{hao} v \iff$$
$$\mu(b_h)v = \mu(b_h)r + \gamma \cdot \sum_o (\mu(b_h)T_{ao}e) \cdot \mu(b_{hao})v.$$

The Poupart and Boutilier (2002) algorithm preserves updates to b_h. If the PSR update rule is:

$$\bar{b}_{hao} = \bar{b}_h \bar{T}_{ao}.$$

then $\forall h \in H_M, b_h)F = \bar{b}_h$, if $\bar{b}_\lambda$ is initialized to $b_\lambda F$.

If $u = e$, as in Littman et al. (2001), the normalized update rule for the vector of test probabilities which make up the belief state is:

$$\mu(\bar{b}_{hao}) = \frac{\mu(\bar{b}_h)\bar{T}_{ao}}{\mu(\bar{b}_h)\bar{T}_{ao}\bar{u}^T}.$$

Littman et al. (2001) have shown that the updated test probabilities are accurate, that is $\forall h \in H_M, \mu(()b_h)F = \mu(\bar{b}_h)$, if $\bar{b}_\lambda$ is initialized to $b_\lambda F$.

In the case in which u is based on the reward function (Poupart and Boutilier, 2002), the value function for the POMDP is preserved, although it may not be possible to properly normalize the test probabilities at each step.

While both of these methods can be used to compress the belief state of the POMDP M, neither takes advantage of observation abstraction.

3.3 PSR Homomorphisms

This section addresses the following question:

- If the input model M were a PSR, could the krylov basis be used to form an abstract shadow model?

In fact, the shadow model derived from a PSR, has a non-linear transition update equation. Unfortunately, both the Shadow Model and Shadow Compatibility abstraction tests rely on the linearity of the shadow model update equation.

To understand how one would construct a shadow PSR from an original PSR, we'll look in a little more depth at how the shadow model transition function/matrix is defined. Take some test $q \in Q$ for the original PSR M. Each update from one history h to the next ($P(q \mid h)$ to $P(q' \mid hao)$) can be broken down into stages. A POMDP has a two stage update: first $P(s' \mid ha)$, then $P(s' \mid hao)$. A PSR lumps both of these steps into a single update operation. The abstract and shadow models instead break the update into three stages:

1. calculate: $P(q'|ha)$
2. update 1 to get: $P(q'|ha\kappa(o))$
3. update 2 to get: $P(q'|hao)$

The abstract model includes update 2, but not update 3. The shadow model, on the other hand, must include update 3, but not update 2. This means we must be able to explicitly

separate the update due to the abstract observation label from the update due to the full observation label.

$$P(q' \mid hao) = \overbrace{P(q' \mid ha)}^{a \text{ update}} \cdot \overbrace{\frac{P(q'\kappa(o) \mid ha)}{P(q' \mid ha) \cdot P(\kappa(o) \mid ha)}}^{\kappa(o) \text{ update}} \cdot \overbrace{\frac{P(q'o \mid ha)}{P(o \mid ha)} \cdot \frac{P(\kappa(o) \mid ha)}{P(q'\kappa(o) \mid ha)}}^{o \text{ update}}$$

The abstract model updates leave off the last update:

$$\begin{aligned} P(q' \mid ha\kappa(o)) &= \overbrace{P(q' \mid ha)}^{a \text{ update}} \cdot \overbrace{\frac{P(q'\kappa(o) \mid ha)}{P(q' \mid ha) \cdot P(\kappa(o) \mid ha)}}^{\kappa(o) \text{ update}} \\ &= \frac{P(q'\kappa(o) \mid ha)}{P(\kappa(o) \mid ha)} \end{aligned}$$

The shadow model updates use the other part of the observation function:

$$\begin{aligned} P(q' \mid h\langle a\kappa(o)\rangle o) &\propto \overbrace{P(q' \mid ha)}^{a \text{ update}} \cdot \overbrace{\frac{P(q'o \mid ha)}{P(o \mid ha)} \cdot \frac{P(\kappa(o) \mid ha)}{P(q'\kappa(o) \mid ha)}}^{o \text{ update}} \\ &\propto \overbrace{P(q' \mid ha)}^{a \text{ update}} \cdot \overbrace{\frac{P(q'o \mid ha)}{P(q'\kappa(o) \mid ha)}}^{o \text{ update}}, \end{aligned}$$

since $\frac{P(\kappa(o)|ha)}{P(o|ha)}$ does not depend on q.

If the history h is replaced with the PSR state vector b_h from the original PSR:

$$\begin{aligned} P(q' \mid h\langle a\kappa(o)\rangle o) &= P(q' \mid ha) \cdot \frac{P(q'o \mid ha)}{P(q'\kappa(o) \mid ha)} \\ &= \left(\sum_q \sum_o w(oq' \mid q, a) \cdot b_h(q) \right) \cdot \left(\frac{\sum_q w(oq' \mid q, a) \cdot b_h(q)}{\sum_q \sum_{o' \in [o]_\kappa} w(oq' \mid q, a) \cdot b_h(q)} \right) \end{aligned}$$

Unfortunately, there is no way to cancel out the sums over q in the ratio portion of this update rule. This means that the update rule does not reduce to a linear transformation.

That is, when M is a PSR, $b_{\xi(hao)}$ cannot be calculated via a linear transformation of $b_{\xi(h)}$. Since the Krylov basis techniques are only applicable to linear transformation functions, the shadow model of a PSR is not a good candidate for Krylov basis reduction, though it may be possible to adapt the PSR representation in some way to avoid this problem.

3.4 Outline

The remainder of this chapter is organized to parallel the last chapter. If κ is a mapping function $\kappa : O \rightarrow \bar{O}$ mapping the observations of a POMDP M to abstract observations., the next several sections will present algorithms that:

- Evaluate κ when for all states s, actions a and observations o, $P(o \mid s, a) > 0$.
- Evaluate κ when there are some observation probabilities which are 0.

The difference is that rather than using CMP Homomorphisms to achieve all of these tasks, this section uses the Krylov Basis to construct an abstract PSR $\bar{M}_\chi$. If the algorithms accept κ, then the abstract PSR should make the same predictions about abstract tests that the POMDP does. Since the abstract PSR models may be value-directed, the proofs in this chapter will focus on showing that the value function is preserved, rather than the abstract belief state updates.

The switch to an abstract PSR representation results in two differences in the set of abstract observation mapping functions accepted. The first difference is that abstract PSRs may be value-directed, where the abstract POMDP approach can only accept observation-directed models. However, even when restricted to the observation-directed case, the abstract PSR test can in some cases accept observation mapping functions that the abstract POMDP approach would reject. This appears to be at least partly due to the fact that the POMDP update rule can be divided into two steps (Equations 2.2 and 2.3), whereas the PSR update rule combines both the action and observation updates (Equation 3.1). Mod-

ifying the POMDP abstraction algorithm to combine the action and observation updates results in a closer match between the acceptance patterns of the two algorithms.

The final sections of this Chapter will compare the acceptance sets and experimental running time of the both POMDP and PSR abstraction finding algorithms.

3.5 Shadow Model Test

In this section it is assumed that for all states s, actions a and observations o:

$$P(o \mid s, a) > 0 \tag{3.7}$$

Given a POMDP $M = (S, A, T, O, \Omega)$ and observation mapping function $\kappa : O \to \bar{O}$, Procedure 3.5.1 tests κ by constructing abstract and shadow PSRs for κ. If the abstract PSR is independent of the shadow PSR, the test succeeds and κ is accepted. Otherwise, κ is rejected. In order to define the shadow and abstract model transition matrices for M, the transition matrices $\{T_{ao}\}$ must be separated into their action and observation portions. For any action a and observation o, observation matrix P_{ao} is a diagonal matrix, where:

$$P_{ao}(i, i) = P(o \mid s_i, a). \tag{3.8}$$

The separate action update matrix T_a has entries:

$$T_a(i, j) = P(s_j \mid s_i, a). \tag{3.9}$$

The combined update matrix is then:

$$T_{ao} = T_a P_{ao}$$

Recall that the POMDP M_χ (Equation 2.14) is defined as $(S, A, T, \bar{O}, \Omega_\chi)$. The set $\{T_{\chi(ao)}\}$ of transition matrices for this POMDP can be defined:

$$T_{\chi(ao)} = T_a P_{\chi(ao)}.$$

where

$$P_{\chi(ao)} = \sum_{o' \in [o]_\kappa} P_{ao}.$$

Recall that the shadow POMDP M_ξ (Equation 2.19) is defined as $(S, A, T, \bar{O}, \Omega_\xi)$. The set $\{T_{\xi(ao)}\}$ of transition matrices for this POMDP can be defined:

$$T_{\xi(ao)} = T_a \cdot \frac{P_{ao}}{P_{\chi(ao)}}$$

where the division symbol indicates entry-wise division (so that $\frac{P_{ao}}{P_{\chi(ao)}}(i,i) = \frac{P_{ao}(i,i)}{P_{\chi(ao)}(i,i)}$). Non-diagonal entries are 0.

Given the assumption in Equation 3.7, for any $a \in A$ and $\bar{o} \in \bar{O}$:

$$\sum_{o \in \bar{o}} \frac{P_{ao}}{P_{a\bar{o}}} = I \tag{3.10}$$

where I is an identity matrix. Procedure 3.5.1 evaluates a given observation mapping κ under this assumption.

The first step that Procedure 3.5.1 takes is the construction of the starting vector for the Krylov Basis. The start vector u for the abstract model can be a reward vector r (value-directed) or e (observation-directed). If the model is to be observation-directed, the observations must first be tested so that $\bar{\zeta}(\kappa(o)) = \zeta(o)$. The next step of the algorithm is to

Procedure 3.5.1 PSR Shadow Model Test

// Initialize the start vector
if value-directed model **then**
  $u = r$
else if output-directed model **then**
  $u = \zeta$
else if observation-directed model **then**
  if $\exists o \in O, \bar{\zeta}(\kappa(o)) \neq \zeta(o)$ **then**
    return false
    $u = e$

// Construct the abstract PSR
$F_\chi \Leftarrow$ KrylovBasis(u, $\{T_{\chi(ao)}\}$)

// Construct the shadow PSR
$F_\xi \Leftarrow$ KrylovBasis(e, $\{T_{\xi(ao)}\}$)

// Construct joint projection matrix F
$F(i, \langle k, l \rangle) = F_\chi(i, k) \cdot F_\xi(i, l)$

// Test the independence of the abstract and shadow PSRs
if $b_\lambda F = \bar{b}_{\chi(\lambda)} \otimes \tilde{b}_{\xi(\lambda)}$ Equation 3.18 **then**
  if $T_{ao} F = \left(\bar{T}_{\chi(ao)} \otimes \tilde{T}_{\xi(ao)} \right) F$ Equation 3.19 **then**
    return F_χ
return false

construct krylov basis projection matrix for the abstract PSR, F_χ. The abstract PSR $\bar{M}_\chi$ is defined as:

$$\bar{M}_\chi = (\bar{Q}, A, \bar{O}, \{\bar{T}_{\chi(ao)}\}, \bar{b}_\lambda) \tag{3.11}$$

where $\bar{Q}$ is the set of tests corresponding to the columns of F_χ, and $\bar{b}_\lambda = b_\lambda F_\chi$. For each action a and observation o, $\bar{T}_{\chi(ao)}$ is the solution to:

$$F_\chi \bar{T}_{\chi(ao)} = T_{\chi(ao)} F_\chi \tag{3.12}$$

from Equation 3.3. From Equation 3.4:

$$F_\chi \bar{u} = u \tag{3.13}$$

Theorem 3.1 will show that $\bar{M}_\chi$ is an accurate compressed model for M_χ, although it has not yet been shown to be an accurate abstraction for M.

Theorem 3.1. $\forall h \in H_M, \bar{b}_{\chi(h)} = b_{\chi(h)} F_\chi$, *where* $b_{\chi(h)}$ *is the un-normalized belief state for* M_χ *(Equation 3.6).*

Proof. Structural Induction.

Base case: $h = \lambda$. By definition:

$$\bar{b}_{\chi(\lambda)} = b_{\chi(\lambda)} F_\chi$$

Inductive step: h to hao. Assume that $\bar{b}_{\chi(h)} = b_{\chi(h)} F_\chi$.

$$\begin{aligned}
b_{\chi(hao)} F_\chi &= b_{\chi(h)} T_{\chi(ao)} F_\chi && \text{Equation 3.6} \\
&= b_{\chi(h)} F_\chi \bar{T}_{\chi(ao)} && \text{Equation 3.12} \\
&= \bar{b}_{\chi(h)} \bar{T}_{\chi(ao)} && \text{Inductive Assumption} \\
&= \bar{b}_{\chi(hao)}
\end{aligned}$$

□

The next step that Procedure 3.5.1 takes is the construction of the abstract shadow PSR. The start vector for the shadow PSR must be e (the shadow model PSR must be observation-directed). If the krylov basis projection matrix for the abstract PSR is denoted F_ξ, then the abstract PSR $\bar{M}_\xi$ is defined as:

$$\bar{M}_\xi = (\tilde{Q}, A, O, \{\tilde{T}_{\xi(ao)}\}, \tilde{b}_\lambda)$$

where $\tilde{Q}$ is the set of tests corresponding to the columns of F_ξ, and $\tilde{b}_\lambda = b_\lambda F_\xi$. For each action a and observation o, $\tilde{T}_{\xi(ao)}$ is the solution to:

$$F_\xi \tilde{T}_{\xi(ao)} = T_{\xi(ao)} F_\xi \tag{3.14}$$

from Equation 3.3. From Equation 3.4:

$$F_\xi \tilde{e} = e \tag{3.15}$$

Theorem 3.2. $\forall h \in H_M, \tilde{b}_{\xi(h)} = b_{\xi(h)} F_\xi$

Proof. Structural Induction.

Base case: $h = \lambda$. By definition:

$$\tilde{b}_{\xi(\lambda)} = b_{\xi(\lambda)} F_\xi$$

Inductive step: h to hao. Assume that $\tilde{b}_{\xi(h)} = b_{\xi(h)} F_\xi$.

$$\begin{aligned}
b_{\xi(hao)} F_\xi &= b_{\xi(h)} T_{\xi(ao)} F_\xi \\
&= b_{\xi(h)} F_\xi \tilde{T}_{\xi(ao)} && \text{Equation 3.14} \\
&= \tilde{b}_{\xi(h)} \tilde{T}_{\xi(ao)} && \text{Inductive Assumption} \\
&= \tilde{b}_{\xi(hao)}
\end{aligned}$$

□

After constructing the abstract PSR and shadow PSR, Procedure 3.5.1 tests the two PSRs to determine whether they are independent. This require the construction of a matrix that projects states in S onto the joint predictions of pairs of tests in $\bar{Q}$ and $\tilde{Q}$. Define the

matrix F as a projection matrix from belief vectors onto the joint abstract & shadow core tests:

$$F(i, \langle k, l \rangle) = F_\chi(i, k) \cdot F_\xi(i, l) \tag{3.16}$$

where $\langle i, j \rangle$ is an index into the columns of F. If F_χ is an $n \times m$ matrix, then $\langle i, j \rangle = i \cdot n + j$.

Theorem 3.3. *In order to convert between F and F_χ:*

$$F(I \otimes \tilde{e}^T) = F_\chi \tag{3.17}$$

where I is an appropriately sized identity matrix, and $\otimes$ denotes the Kronecker Product of two matrices.

Proof.

$$\begin{aligned}
F(I \otimes \tilde{e}^T)(i, j) &= \sum_{\langle k,l \rangle} F(i, \langle k, l \rangle) \cdot (I \otimes \tilde{e}^T)(\langle k, l \rangle, j) \\
&= \sum_{k,l} F_\chi(i, k) \cdot F_\xi(i, l) \cdot I(k, j) \cdot \tilde{e}(l) && \text{Kronecker Product} \\
&= \sum_{l} F_\chi(i, j) \cdot F_\xi(i, l) \cdot I(j, j) \cdot \tilde{e}(l) && I(k, j) = 0 \text{ if } k \neq j \\
&= F_\chi(i, j) \cdot \sum_{l} F_\xi(i, l) \cdot \tilde{e}(l) \\
&= F_\chi(i, j) \cdot (F_\xi \tilde{e})(i) \\
&= F_\chi(i, j) \cdot e(i) && \text{Equation 3.15} \\
&= F_\chi(i, j)
\end{aligned}$$

□

The following tests are the shadow model constraints:

$$b_\lambda F = \bar{b}_{\chi(\lambda)} \otimes \tilde{b}_{\xi(\lambda)} \tag{3.18}$$

$$T_{ao} F = F \left(\bar{T}_{\chi(ao)} \otimes \tilde{T}_{\xi(ao)} \right) \tag{3.19}$$

if these test fail, then the shadow and abstract models may be correlated, and the test fails.

The following theorem states that if Equations 3.18 and 3.19 are satisfied, then the belief state factors into abstract and shadow components for every h. This will be used to show that these two tests are sufficient to show that the value function for r (or ζ) is preserved, in Theorem 3.7.

Theorem 3.4. *If M is PSR, and $\bar{M}$ and $\tilde{M}$ are abstract and shadow models which satisfy Equations 3.18 & 3.19, then for all $h \in H_M$, $Fb_h = \bar{b}_{\chi(h)} \otimes \tilde{b}_{\xi(h)}$.*

Proof. By Induction on h.

Base case ($h = \lambda$): Equation 3.18.

Inductive assumption:

$$b_h F = \bar{b}_{\chi(h)} \otimes \tilde{b}_{\xi(h)}$$

Inductive step (h to hao):

$$\begin{aligned} b_h T_{ao} F &= b_h F \left(\bar{T}_{\chi(ao)} \otimes \tilde{T}_{\xi(ao)} \right) \\ &= \left(\bar{b}_{\chi(h)} \otimes \tilde{b}_{\xi(h)} \right) \left(\bar{T}_{\chi(ao)} \otimes \tilde{T}_{\xi(ao)} \right) \\ &= \left(\bar{b}_{\chi(h)} \bar{T}_{\chi(ao)} \right) \otimes \left(\tilde{b}_{\xi(h)} \tilde{T}_{\xi(ao)} \right) \\ &= \bar{b}_{\chi(hao)} \otimes \tilde{b}_{\xi(hao)} \end{aligned}$$

□

The following corollary to this theorem will be helpful in proving that the abstract value function is accurate (Theorem 3.7).

Corollary 3.5. *Theorem 3.4 implies that for any $h \in H_M$, $b_h F(I \otimes \tilde{e}^T) = \bar{b}_{\chi(h)} \cdot c_h$, where c_h is a scalar constant, and $c_h = (\tilde{b}_{\xi(h)} \tilde{e}^T)$.*

Proof.

$$
\begin{aligned}
b_h F(I \otimes \tilde{e}^T) &= (\bar{b}_{\chi(h)} \otimes \tilde{b}_{\xi(h)})(I \otimes \tilde{e}^T) && \text{Theorem 3.4} \\
&= (\bar{b}_{\chi(h)} I) \otimes (\tilde{b}_{\xi(h)} \tilde{e}^T) \\
&= \bar{b}_{\chi(h)} \cdot (\tilde{b}_{\xi(h)} \tilde{e}^T) && \text{since } \tilde{b}_{\xi(h)} \tilde{e}^T \text{ is a scalar} \\
&= \bar{b}_{\chi(h)} \cdot c_h
\end{aligned}
$$

□

The following Lemma will also be helpful in proving that the abstract value function can be lifted back to the original POMDP. It relies on the fact that abstract observations are treated like actions in the shadow model. Therefore, for each abstract observation $\bar{o} \in \bar{O}$, the observations within $\bar{o}$ are normalized to sum to one. That is, $\sum_{o \in \bar{o}} \frac{P_o(i,i)}{P_{\bar{o}}(i,i)} = 1$.

Lemma 3.6. *Equation 3.7 implies that for any $h \in H_M$, and $\bar{o} \in \bar{O}$, $\sum_{o \in \bar{o}} \tilde{b}_{\xi(hao)} \tilde{e}^T = \tilde{b}_{\xi(h)} \tilde{e}^T$. Equivalently, $\sum_{o \in \bar{o}} c_{hao} = c_h$.*

Proof.

$$
\begin{aligned}
\sum_{o\in\bar{o}} \tilde{b}_{\xi(hao)}\tilde{e}^T &= \sum_{o\in\bar{o}} \tilde{b}_{\xi(h)}\tilde{T}_{\xi(ao)}\tilde{e}^T \\
&= \sum_{o\in\bar{o}} b_{\xi(h)}T_a\frac{P_{ao}}{P_{a\bar{o}}}e^T && \text{Theorem 3.2} \\
&= b_{\xi(h)}T_a\left(\sum_{o\in\bar{o}}\frac{P_{ao}}{P_{a\bar{o}}}\right)e^T \\
&= b_{\xi(h)}T_aIe^T && \text{Equation 3.10} \\
&= b_{\xi(h)}T_ae^T \\
&= b_{\xi(h)}e^T && \text{each row of } T_a \text{ sums to one.} \\
&= \tilde{b}_{\xi(h)}\tilde{e}^T && \text{Theorem 3.2}
\end{aligned}
$$

□

Theorem 3.7. *If Equations 3.18 and 3.19 are satisfied, and thus Theorem 3.4 is true, then the value function for r can be lifted from the abstract PSR $\bar{M}_\chi$ to the original POMDP M.*

Proof. Define $v = F_\chi\bar{v}$.

$$
\begin{aligned}
b_hv = b_hr + \gamma\cdot\sum_o b_hT_{ao}v \iff \\
b_hF_\chi\bar{v} = b_hF_\chi\bar{r} + \gamma\cdot\sum_{o\in O} b_{hao}F_\chi\bar{v} \iff && \text{(Definition of } v\text{)} \\
b_hF(I\otimes\tilde{e})\bar{v} = b_hF(I\otimes\tilde{e})\bar{r} + \gamma\cdot\sum_{o\in O} b_{hao}F(I\otimes\tilde{e})\bar{v} \iff && \text{(Equation 3.17)} \\
\bar{b}_{\chi(h)}c_h\bar{v} = \bar{b}_{\chi(h)}c_h\bar{r} + \gamma\cdot\sum_{o\in O}\bar{b}_{\chi(hao)}c_{hao}\bar{v} \iff && \text{(Corollary 3.5)} \\
\bar{b}_{\chi(h)}\bar{v}c_h = \bar{b}_{\chi(h)}\bar{r}c_h + \gamma\cdot\sum_{\bar{o}\in\bar{O}}\bar{b}_{\chi(ha)\bar{o}}\bar{v}\sum_{o\in\bar{o}}c_{hao} \iff && (\kappa \text{ partitions } O) \\
\bar{b}_{\chi(h)}\bar{v}c_h = \bar{b}_{\chi(h)}\bar{r}c_h + \gamma\cdot\sum_{\bar{o}\in\bar{O}}\bar{b}_{\chi(ha)\bar{o}}\bar{v}c_h \iff && \text{(Lemma 3.6)} \\
\bar{b}_{\chi(h)}\bar{v} = \bar{b}_{\chi(h)}\bar{r} + \gamma\cdot\sum_{\bar{o}\in\bar{O}}\bar{b}_{\chi(ha)\bar{o}}\bar{v} && \text{(Cancel } c_h\text{)}
\end{aligned}
$$

□

3.6 Compatibility Test

The proofs of the last section relied on the fact that Equation 3.7 was satisfied in M. However, as the last chapter has shown (Section 2.4.7), this is not always the case. This section will define state "compatibility" criteria in the PSR framework in a similar manner to Section 2.5.

As in Section 2.5, there are several important models in the compatibility test.

- M is the POMDP
- M_ξ is the shadow POMDP, defined in Equation 2.19
- $\bar{M}_\chi$ is the abstract PSR, defined in Equation 3.11
- $\bar{M}_\eta$ is the availability PSR, from the availability POMDP M_η (Equation 2.27)

The POMDP M_η has the following observation "availability" matrices, for each action a and observation o:

$$P_{\eta(ao)} = \sum_{o' \in [o]_\kappa} P_{\xi(ao')}. \tag{3.20}$$

Each diagonal entry $P_{\eta(ao)}(i,i)$ is equivalent tot he availability function $\eta(\bar{o}, s, a)$ (Equation 2.26). The transition matrices for M_η are:

$$T_{\eta(ao)} = T_a \cdot P_{\eta(o)} \tag{3.21}$$

The abstract PSR $\bar{M}_\eta$ can be constructed by finding the Krylov Basis $Kr(\{T_{\eta(ao)}, e\}$, where e is a vector of ones. The availability PSR, like the abstract shadow PSR, must be

observation-directed. If F_η is the projection matrix returned by $Kr(\{T_{\eta(ao)}, e\}$, then $\bar{M}_\chi$ is defined as:

$$\bar{M}_\eta = (\bar{Q}, A, \bar{O}, \{\bar{T}_{\eta(ao)}\}, \bar{b}_\lambda) \tag{3.22}$$

where $\bar{Q}$ is the set of tests corresponding to the columns of F_η, and $\bar{b}_\lambda = b_\lambda F_\eta$. For each action a and observation o, $\bar{T}_{\eta(ao)}$ is the solution to:

$$\bar{T}_{\eta(ao)} F_\eta = F_\eta T_{\eta(ao)} \tag{3.23}$$

from Equation 3.3. From Equation 3.4:

$$\bar{e}_\eta F_\eta = e \tag{3.24}$$

Let I_c be a diagonal compatibility function matrix, in which $I_c(\langle i, j\rangle, \langle i, j\rangle) = 1$ if and only if i and j are compatible, and all other entries are 0. The compatibility algorithm (Procedure 3.7.2) constructs I_c, but first we will examine the properties it must have.

For each $a \in A$ and $o \in O$, there must exist a weight matrix W_{ao}, such that:

$$I_c W_{ao} I_c (F_\chi \otimes I) = I_c (T_{\chi(ao)} \otimes T_{\xi(ao)})(F_\chi \otimes I) \tag{3.25}$$

$$I_c W_{ao} I_c (I \otimes F_\eta) = I_c (T_{ao} \otimes T_{\eta(ao)})(I \otimes F_\eta) \tag{3.26}$$

where I is always an appropriately sized identity matrix, and $\otimes$ denotes the Kronecker product of two matrices.

There must also be a vector w_λ for the initial belief state such that:

$$w_\lambda I_c (I \otimes F_\eta) = b_\lambda \otimes \bar{b}_{\eta(\lambda)} \tag{3.27}$$

$$w_\lambda I_c (F_\chi \otimes I) = b_{\chi(\lambda)} \otimes b_{\xi(\lambda)} \tag{3.28}$$

Equations 3.25 - 3.28 are the PSR shadow compatibility model acceptance constraints. Next, these constraints will be shown to imply that the value function is preserved in the abstract model.

If $\bar{v}$ is the value function for $\bar{M}_\chi$ and the value function for M is defined as $v = F_\chi \bar{v}$, then

$$b_h v = b_h r + \gamma \sum_{o \in O} b_{hao} v \iff \bar{b}_{\chi(h)} \bar{v} = \bar{b}_{\chi(h)} \bar{r} + \gamma \sum_{o \in O} \bar{b}_{\chi(ha)\bar{o}} \bar{v}$$

as Theorem 3.8 will show.

In order to show that Equations 3.25 - 3.28 are sufficient criteria for correctness, the hypothetical POMDP model $\breve{M}$ will be examined. $\breve{M}$ does not actually need to be constructed. Its only purpose is to show that the tests in Equations 3.18 and 3.19 are rigorous. The POMDP $\breve{M}$ is defined as follows:

$$\breve{M} = (\breve{S}, A, O, \{\breve{T}_{ao}\}, \breve{b}_\lambda)$$

where $\breve{S} = S \times S$, and for each action a and observation o:

$$\{\breve{T}_{ao} = I_c W_{ao} I_c\}. \tag{3.29}$$

The initial belief for $\breve{M}$ is $\breve{b}_\lambda = w_\lambda I_c$.

Theorem 3.8. *From Equations 3.26, 3.25, 3.27 & 3.28, for every $h \in H_M$:*

$$\breve{b}_h (I \otimes F_\eta) = b_h \otimes \bar{b}_{\eta(h)} \tag{3.30}$$

$$\breve{b}_h (F_\chi \otimes I) = \bar{b}_{\chi(h)} \otimes b_{\xi(h)} \tag{3.31}$$

where $\breve{b}_h(\langle i, j \rangle) = 0$ if i and j are incompatible — that is, $\breve{b}_h I_c = \breve{b}_h$.

Proof. By Structural Induction.

Base case: $h = \lambda$. Equations 3.27 & 3.28. In addition, since $\breve{b}_\lambda = w_\lambda I_c$:

$$\begin{aligned}\breve{b}_\lambda I_c &= w_\lambda I_c I_c \\ &= w_\lambda I_c \\ &= \breve{b}_\lambda\end{aligned}$$

Inductive step: h to hao.

Equation 3.30:

$$\begin{aligned}\breve{b}_{hao}(I \otimes F_\eta) &= \breve{b}_h \breve{T}_{ao}(I \otimes F_\eta) \\ &= \breve{b}_h I_c W_{ao} I_c (I \otimes F_\eta) && \text{Equation 3.29} \\ &= \breve{b}_h I_c (T_{ao} \otimes F_\eta \bar{T}_{\eta(ao)}) && \text{Equation 3.26} \\ &= \breve{b}_h (T_{ao} \otimes F_\eta \bar{T}_{\eta(ao)}) && \text{Inductive Assumption} \\ &= \breve{b}_h (I \otimes F_\eta)(T_{ao} \otimes \bar{T}_{\eta(ao)}) && \text{Kronecker Product} \\ &= b_h \otimes \bar{b}_{\eta(h)})(T_{ao} \otimes \bar{T}_{\eta(ao)}) && \text{Inductive Assumption} \\ &= b_{hao} \otimes \bar{b}_{\eta(hao)} && \text{Kronecker Product}\end{aligned}$$

Equation 3.31:

$$\begin{aligned}\breve{b}_{hao}(F_\chi \otimes I) &= \breve{b}_h \breve{T}_{ao}(F_\chi \otimes I) \\ &= \breve{b}_h I_c (F_\chi \bar{T}_{\chi(ao)} \otimes T_{ao}) && \text{Equation 3.25} \\ &= \breve{b}_h (F_\chi \bar{T}_{\chi(ao)} \otimes T_{ao}) && \text{Inductive Assumption} \\ &= \breve{b}_h (F_\chi \otimes I)(\bar{T}_{\chi(ao)} \otimes T_{ao}) && \text{Kronecker Product} \\ &= (\bar{b}_{\chi(h)} \otimes b_h)(\bar{T}_{\chi(ao)} \otimes T_{ao}) && \text{Inductive Assumption} \\ &= \bar{b}_{\chi(hao)} \otimes b_{hao} && \text{Kronecker Product}\end{aligned}$$

□

Now we must show that this implies that the abstract PSR model $\bar{M}_\chi$ is an accurate reduction of M.

First, the belief state and abstract belief state are related through a constant, at each history $c_h = \frac{b_{\xi(h)}e^T}{\bar{b}_{\eta(h)}\bar{e}_\eta^T}$.

Lemma 3.9. *For every h in H_M, $b_h F_\chi = \bar{b}_{\chi(h)} \cdot \frac{b_{\xi(h)}e^T}{\bar{b}_{\eta(h)}\bar{e}_\eta^T} = \bar{b}_{\chi(h)} \cdot c_h$*

Proof. Show, equivalently, that $b_h F_\chi \cdot \bar{b}_{\eta(h)}\bar{e}_\eta^T = \bar{b}_{\chi(h)} \cdot b_{\xi(h)}e^T$, by showing that both sides of the equation are equivalent to $\breve{b}_h(F_\chi \otimes e^T)$.

Part 1: $\breve{b}_h(F_\chi \otimes e^T) = b_h F_\chi \cdot \bar{b}_{\eta(h)}\bar{e}_\eta^T$

$$\begin{aligned}
\breve{b}_h(F_\chi \otimes e^T) &= \breve{b}_h(I \otimes F_\eta)(F_\chi \otimes \bar{e}_\eta^T) && \text{Kronecker Product} \\
&= (b_h \otimes \bar{b}_{\eta(h)})(F_\chi \otimes \bar{e}_\eta^T) && \text{Theorem 3.8} \\
&= b_h F_\chi \otimes \bar{b}_{\eta(h)}\bar{e}_\eta^T && \text{Kronecker Product}
\end{aligned}$$

Part 2: $\breve{b}_h(F_\chi \otimes e^T) = \bar{b}_{\chi(h)} \cdot b_{\xi(h)}e^T$

$$\begin{aligned}
\breve{b}_h(F_\chi \otimes \bar{e}_\eta^T) &= \breve{b}_h(F_\chi \otimes I)(I \otimes e^T) && \text{Kronecker Product} \\
&= (\bar{b}_{\chi(h)} \otimes b_{\xi(h)})(I \otimes e^T) && \text{Theorem 3.8} \\
&= \bar{b}_{\chi(h)} \cdot b_{\xi(h)}e^T && \text{Kronecker Product}
\end{aligned}$$

Combined, we have:

$$\begin{aligned}
\breve{b}_h(F_\chi \otimes e^T) &= \breve{b}_h(F_\chi \otimes e^T) \\
b_h F_\chi \cdot \bar{b}_{\eta(h)}\bar{e}_\eta^T &= \bar{b}_{\chi(h)} \cdot b_{\xi(h)}e^T \\
b_h F_\chi &= \bar{b}_{\chi(h)} \cdot \frac{b_{\xi(h)}e^T}{\bar{b}_{\eta(h)}\bar{e}_\eta^T} \\
b_h F_\chi &= \bar{b}_{\chi(h)} \cdot c_h
\end{aligned}$$

□

There are actually two equivalent ways of defining the constant c_h.

Lemma 3.10. $c_h = \frac{b_h F_\chi e^T}{\bar{b}_{\chi(h)} e^T} = \frac{b_{\xi(h)} e^T}{\bar{b}_{\eta(h)} \bar{e}_\eta^T}$ *where* e *is a vector of ones.*

Proof.

$$
\begin{aligned}
b_h F_\chi &= \frac{b_{\xi(h)} e^T}{\bar{b}_{\eta(h)} \bar{e}_\eta^T} \cdot \bar{b}_{\chi(h)} && \text{Lemma 3.9} \\
b_h F_\chi e^T &= \frac{b_{\xi(h)} e^T}{\bar{b}_{\eta(h)} \bar{e}_\eta^T} \cdot \bar{b}_{\chi(h)} e^T \\
\frac{b_h F_\chi e^T}{\bar{b}_{\chi(h)} e^T} &= \frac{b_{\xi(h)} e^T}{\bar{b}_{\eta(h)} \bar{e}_\eta^T}
\end{aligned}
$$

□

Here $b_h F_\chi e^T$ is simply the sum of the elements in $b_h F_\chi$ — it is not equivalent to $b_h e^T$. Since the abstract model may start with the reward as the initial vector ($u = r$), there may not be any vector $\bar{e}$ that solves $e^T = F_\chi \bar{e}^T$, and thus, no vector that solves $b_h F_\chi \bar{e}^T = b_h e^T$. Nonetheless, since each element $b_h F_\chi(i)$ is a constant multiple of $\bar{b}_{\chi(h)}(i)$, the constant c_h can be recovered without $\bar{e}$.

Lemma 3.11. *Given Lemmas 3.9 and 3.10, for any* $\bar{o} \in \bar{O}$, $\sum_{o \in \bar{o}} c_{hao} = c_h$.

Proof.

$$
\begin{aligned}
\sum_{o\in\bar{o}} c_{hao} &= \sum_{o\in\bar{o}} \frac{b_{hao}F_\chi e^T}{\bar{b}_{\chi(hao)}e^T} && \text{Lemma 3.10}\\
&= \frac{b_h \sum_{o\in\bar{o}} T_{ao}F_\chi e^T}{\bar{b}_{\chi(hao)}e^T}\\
&= \frac{b_h T_{\chi(ao)}F_\chi e^T}{\bar{b}_{\chi(hao)}e^T} && \text{Equation}\\
&= \frac{b_h F_\chi \bar{T}_{\chi(ao)}e^T}{\bar{b}_{\chi(hao)}e^T} && \text{Equation 3.14}\\
&= \frac{c_h \cdot \bar{b}_{\chi(h)}\bar{T}_{\chi(ao)}e^T}{\bar{b}_{\chi(hao)}e^T} && \text{Lemma 3.9}\\
&= c_h \cdot \frac{\bar{b}_{\chi(hao)}e^T}{\bar{b}_{\chi(hao)}e^T}\\
&= c_h
\end{aligned}
$$

□

Finally, we can show that the value function lifts from the abstract model to the true model.

Theorem 3.12.

Proof. Define $v = F_\chi \bar{v}$.

$$
\begin{aligned}
b_h v = b_h r + \gamma \sum_{o\in O} b_{hao} v \iff & \\
b_h F_\chi \bar{v} = b_h F_\chi \bar{r} + \gamma \sum_{o\in O} b_{hao} F_\chi \bar{v} \iff & && \text{Definition of } v\\
c_h \cdot \bar{b}_{\chi(h)}\bar{v} = c_h \cdot \bar{b}_{\chi(h)}\bar{r} + \gamma \sum_{o\in O} c_{hao} \cdot \bar{b}_{\chi(hao)}\bar{v} \iff & && \text{Lemma 3.9}\\
c_h \cdot \bar{b}_{\chi(h)}\bar{v} = c_h \cdot \bar{b}_{\chi(h)}\bar{r} + \gamma \sum_{o\in O} \bar{b}_{\chi(ha)\bar{o}}\bar{v} \sum_{o\in\bar{o}} c_{hao} \iff & \\
c_h \cdot \bar{b}_{\chi(h)}\bar{v} = c_h \cdot \bar{b}_{\chi(h)}\bar{r} + \gamma \sum_{o\in O} \bar{b}_{\chi(ha)\bar{o}}\bar{v} c_h \iff & && \text{Lemma 3.11}\\
\bar{b}_{\chi(h)}\bar{v} = \bar{b}_{\chi(h)}\bar{r} + \gamma \sum_{o\in O} \bar{b}_{\chi(ha)\bar{o}}\bar{v} && \text{Cancel constants}
\end{aligned}
$$

□

Procedure 3.7.1 PSR Compatibility Check(M, κ)

```
// Initialize the start vector
if value-directed model then
   u = r
else if observation-directed model then
   if ∃o ∈ O, ζ̄(κ(o)) ≠ ζ(o) then
      return false
      u = e

// Construct the abstract PSR
F_χ ⇐ KrylovBasis(u, {T_χ(ao)})

// Construct the shadow PSR
F_η ⇐ KrylovBasis(e, {T_η(ao)})

I_c ⇐ psrCompatibilityMatrix(M, F_χ, F_η)

return checkInitialBelief(b_λ, I_c, F_χ, F_η)
```

3.7 Compatibility Algorithm

Procedure 3.7.1 implements the PSR based compatibility check for the POMDP M and observation mapping function κ. It defines the projection matrices F_χ and F_η for the PSRs $\bar{M}_\chi$ (Equation 3.11) and $\bar{M}_\eta$ (Equation 3.22). The next step constructs the compatibility matrix I_c, using Procedure 3.7.2. Finally, if Procedure 3.7.3 succeeds for the initial belief vector, the procedure returns true, otherwise, it returns false and rejects κ.

Procedure 3.7.2 constructs the compatibility matrix I_c. I_c must be constructed such that for each $a \in A$ and $o \in O$, there exists a weight matrix W_{ao}, such that Equations 3.26 and 3.25 are satisfied. This constrains the entries of I_c. If there is no solution that would satisfy the constraints for a particular row of W_{ao}, I_c for that row must be 0.

Procedure 3.7.2 determines which rows of W_{ao} have solutions. Solvable rows correspond to compatible pairs of states. At the end of each iteration, the diagonal entries of I_c are 1 for each row of W_{ao} for which a solution exists, and 0 for all other rows. This is achieved by solving a series of linear equations.

Procedure 3.7.2 psrCompatibilityMatrix(M, F_χ, F_η

$I_c \Leftarrow I$ // identity matrix
repeat
$I_{old} \Leftarrow I_c$

// Abstract projection matrix (from Equation 3.25)
$abstractProjection \Leftarrow I_c(F_\chi \otimes I)$
// Availability projection matrix (from Equation 3.26)
$availabilityProjection \Leftarrow I_c(I \otimes F_\eta)$
$A \Leftarrow [abstractProjection : availabilityProjection]$

for all $a \in A, o \in O$ **do**
// Abstract prediction matrix (from Equation 3.25)
$abstractPrediction \Leftarrow (T_{ao} \otimes T_{ao})(F_\chi \otimes I)$
// Availability prediction matrix (from Equation 3.26)
$availabilityPrediction \Leftarrow (T_{ao} \otimes T_{ao})(I \otimes F_\eta)$
$Y \Leftarrow [abstractPrediction : availabilityPrediction]$

for all $i, j \in S$ **do**
// Solve the system of equations for the $\langle i, j \rangle^{th}$ row of W_{ao}
$y \Leftarrow Y(\langle i, j \rangle, \cdot)$ // The $\langle i, j \rangle^{th}$ row
if the solution x to $A^T x = y^T$ exists **then**
$W_{ao}(\langle i, j \rangle, \cdot) \Leftarrow x^T$
else
$I_c(\langle i, j \rangle, \langle i, j \rangle) \Leftarrow 0$
until $I_{old} = I_c$
return I_c

The $\langle i, j \rangle^{th}$ row of W_{ao} can be found, if it exists, by solving the following system of equations for x:

$$xI_{old}(I \otimes F_\eta) = (T_{ao} \otimes T_{\eta(ao)})(I \otimes F_\eta)(\langle i, j \rangle, \cdot)$$

$$xI_{old}(F_\chi \otimes I) = (T_{\chi(ao)} \otimes T_{\xi(ao)})(F_\chi \otimes I)(\langle i, j \rangle, \cdot)$$

where the terms on the right hand side are row vectors of next time step predictions for the selected state pair $\langle i, j \rangle$. If the solution for the row of W_{ao} corresponding to $\langle i, j \rangle$ exists, then the diagonal entry of I_c for this pair remains 1. If no such x exists, the entry for $\langle i, j \rangle$

Procedure 3.7.3 checkInitialBelief($b_\lambda, I_c, F_\chi, F_\eta$)

$A \Leftarrow [I_c(I \otimes F_\eta) : I_c(F_\chi \otimes I)]$
$y \Leftarrow [b_\lambda \otimes \bar{b}_{\eta(\lambda)} : b_{\chi(\lambda)} \otimes b_{\xi(\lambda)}]$
if the equation $xA = y$ has a solution **then**
  return x as $\breve{b}_\lambda$
else
  return false

in I_c is set to 0. Thus, I_c picks out those rows that have solutions (these are the compatible pairs of states), and sets all rows that do not have solutions to 0.

At the end of each iteration of the main repeat loop in Procedure 3.7.2, the following constraints are true for all actions a and observations o:

$$I_c W_{ao} I_{old}(I \otimes F_\eta) = I_c(T_{ao} \otimes T_{\eta(ao)})(I \otimes F_\eta)$$
$$I_c W_{ao} I_{old}(F_\chi \otimes I) = I_c(T_{\chi(ao)} \otimes T_{\xi(ao)})(F_\chi \otimes I)$$

When $I_{old} = I_c$, these become Equations 3.26 and 3.25.

Finally, Procedure 3.7.3 finds a weight vector to satisfy Equations 3.27 and 3.28 by solving for w_λ. If there is a real valued solution, Procedure 3.7.1 succeeds and accepts the abstract model $\bar{M}_\chi$ with compression matrix F_χ and observation mapping function κ. Otherwise, the method fails and rejects the observation mapping function.

3.7.1 Time Analysis

In the worst case, the state compatibility test run time is dominated by the time needed to construct the compatibility function. Procedure 3.7.2 has a total of 5 nested loops. The outer "repeat" loop could execute up to $|S|^2$ times, if each iteration marks only one pairs of states as incompatible, and all pairs of states are incompatible in the end. The nested for loops execute $|A| \cdot |O| \cdot |S|^2$ times.

Within these for loops, the system of equations $A^T x = y^T$ must be solved, where A is a $|S|^2 \times |S|^2$ matrix. This operation has a run time of $O(n^3)$, where n is the dimensionality of the matrix (Cormen et al., 2009), for a total time complexity of $O((|S|^2)^3)$ or $O(|S|^6)$.

The total worst case time complexity is thus $O(|A| \cdot |O| \cdot |S|^{10})$. This differs by a factor of $|O| \cdot |S|$ from the POMDP compatibility algorithm of Section 2.5.3.

3.8 Comparison of PSR and POMDP Methods

This next few sections compare the POMDP compatibility and PSR compatibility approaches, and demonstrate that the PSR compatibility approach accepts some valid models that the POMDP compatibility approach does not. There are three sources of the difference between the two tests, and each will be examined in turn:

1. Abstract PSR models may be value-directed or observation-directed, whereas abstract POMDP models must be observation-directed.

2. The abstract POMDP state predictions must be consistent at two points: after the action update, and after the observation update. The abstract PSR belief vector is only required to be consistent after both updates have been completed.

3. The PSR uses a set of basis tests, rather than state.

The next several sections will focus on explaining these differences between the two tests.

3.9 Observation and Value-directed Models

One of the two differences between the POMDP and PSR tests is that the PSR tests support both value-directed and observation-directed models. Figure 3.3 represents the following POMDP:

States: Each square in Figure 3.3 represents a state.

Actions: up, $down$, $left$, $right$

Transitions: Actions fail with a small probability ϵ. Failure results in no change to the state. The action $right$ is noisy when transitioning to the states with cheese in them, ending in each cheese state with the designated probability.

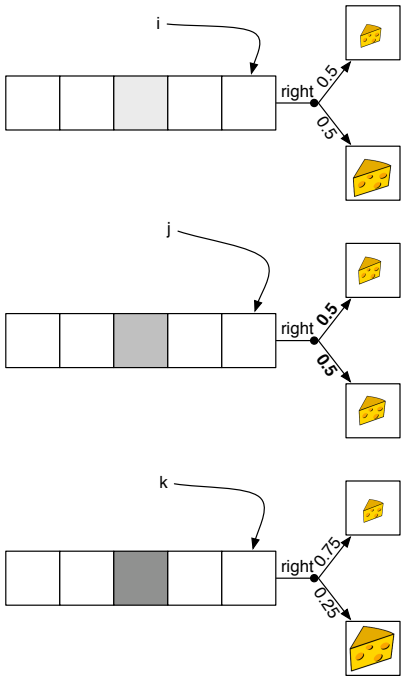


Figure 3.3. Three Hallway domain. Each hallway ends with a transition which has a different reward distribution, but the same mean expected reward.

Observations: $white, lightgrey, grey, black, smallCheese, mediumCheese, largeCheese, hugeCheese$

Observation Function: In each state, the agent deterministically observes the features of the current square.

Initial Belief State: Uniform probability of being in each of the three leftmost states.

Reward Function:

$$smallCheese : 4$$

$$mediumCheese : 6$$

$$largeCheese : 8$$

$$hugeCheese : 12$$

$$\text{all other obserations} : 0$$

Consider the output function ζ corresponding to the reward function for this domain.

While each of the states labeled i j and k transitions has a different pattern of next state transitions for the action $right$, the expected reward received in each state under this action is 6 in each of these states. The value-directed PSR model predicts only this expected reward value, where the POMDP test and observation-directed PSR test predict the exact distribution of ζ at the next time step. This means that the observation distinctions between $lightgrey$, $grey$ and $black$ are necessary for accurate predictions in the observation-directed models. In the value-directed model, these three observations may be clustered into a single abstract observation (see Table 3.1 for test results).

Despite the fact that observation-directed abstract models are often larger than their value-directed counterparts, in some cases, observation-directed models may nonetheless be preferable, as they may be reused over multiple tasks (Wolfe and Barto, 2006).

Domain	Observation Map (κ)	POMDP	PSR-Obs	PSR-Val
Figure 3.3	$\{lg, g, b\}, \{w\}, \{sC, mC, lC, hC\}$			pass
	$\{lg, g, b\}, \{w\}, \{sC\}, \{mC\}, \{lC\}, \{hC\}$	fail	fail	
	$\{lg\}, \{g\}, \{b\}, \{w\}, \{sC\}, \{mC\}, \{lC\}, \{hC\}$	pass	pass	

Table 3.1. Comparison of Observation and Value-directed models.

3.10 PSR vs. POMDP: One Step and Two Step Update Models

Even when the PSR abstraction is constrained to be observation-directed, the two tests differ in the set of observation maps they accept, with the PSR test accepting more observation maps than the POMDP test. The Integer Counter domain (Figure 3.4) represents a binary integer counter. The counter can be advanced by adding one to the integer, or decreased by subtracting one from the integer. The POMDP definition is as follows:

States: A counter with n bits has 2^n states. In the figure, there are 7 bits. The experiments shown use a counter with 5 bits.

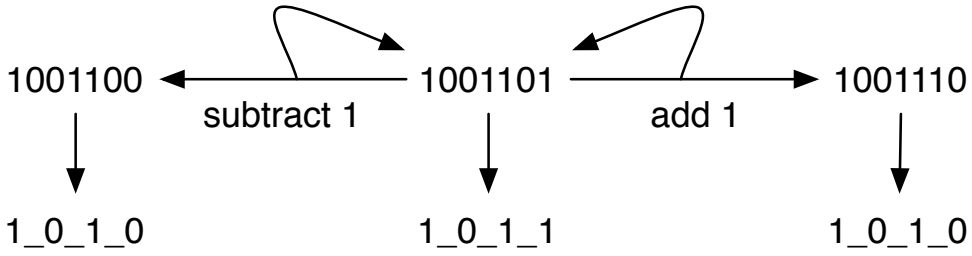


Figure 3.4. Integer Counter Domain. Three states are shown, representing three numbers of 7 digits each. The "add 1" action increases the counter by one, with noisy transitions, and the action "subtract 1" decreases the counter by one, again with noisy transitions. Every other bit is hidden, so that observations include only every other bit.

Actions: $+1, -1$

Transitions: Actions fail with a small probability ϵ. Failure results in no change to the state.

Observations: Every other bit is observed, beginning with the lowest order bit.

Observation Function: Observation probabilities are deterministic.

Initial Belief State: Uniform probability on all states.

Consider the output function where each state s outputs the value $s \mod 2^3 - s \mod 2^2$, or the value of the 3^{rd} bit location. Higher order bits are not useful for predicting the value of this output function, but any lower order bits are. Since only alternate bits are observed, this implies that the observation mapping function $\kappa(o) = o \mod 2^2$ should be self-sufficient. However, the POMDP test rejects this observation map, while the OC-POMDP and PSR tests accept this observation map, as shown in Table 3.2.

In this particular case the difference between the two algorithms does not stem from the difference between using the basis vector vs a belief state vector to represent state. Instead, it stems from the point at which the state vector is calculated. Figure 3.5 illustrates two different bayesian networks that could be used to model a partially observable domain. The

Domain	Observation Map (κ)	POMDP	PSR
Binary Integer Figure 3.4 with 5 bits	$\kappa(o) = o \mod 4$	fail	pass

Table 3.2. Comparison of observation-directed POMDP and PSR algorithms.

POMDP tests described in Chapter 2 were designed based on the type of model shown in Figure 3.5(a). This section defines a modified model and acceptance test, the Observation Conditional POMDP model and test, based on the Bayesian Network shown in Figure 3.5(b).

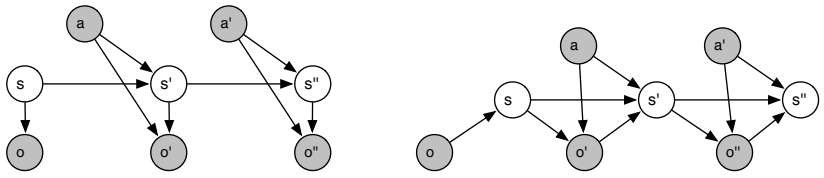
Let $A(POMDP)$ denote the set of M, κ pairs accepted by the POMDP test, and similarly let $A(OC-POMDP)$ and $A(PSR)$ be the accept sets corresponding to the Observation Conditional POMDP and PSR tests. This section will show that:

- $A(POMDP) \subset A(OC-POMDP)$
  - The success of the POMDP test implies the success of the OC-POMDP test
  - There is at least one M, κ pair for which OC-POMDP returns $true$ and POMDP returns $false$.
- $A(OC-POMDP) \subset A(PSR)$
  - The success of the OC-POMDP test implies the success of the PSR test
  - There is at least one M, κ pair for which the PSR test returns $true$ and OC-POMDP returns $false$.

Figure 3.5(a), illustrates the usual POMDP representation, with update equations:

$$b_{hao}(s') \propto P(o \mid s', a) \sum_{s \in S} P(s' \mid s, a) \cdot b_h(s) \tag{3.32}$$

In Figure 3.5(b), the link between each state and its corresponding observation has been reversed, with added dependency links as necessary to preserve the probability distribution



(a) (b)

Figure 3.5. The bayesian model from which the POMDP test is derived (Figure 3.5(a)) and the bayesian model from which the OC-POMDP test is derived (Figure 3.5(b)).

represented by the Bayesian network. This is the type of model by which the OC-POMCP tests are defined. The update equations for this model are:

$$b_{hao}(s') \propto \sum_{s \in S} P(s', o \mid s, a) \cdot b_h(s) \tag{3.33}$$

and the homomorphism constraints for the OC-POMDP tests are as follows. Abstract state constraints:

$$P(f(s'), \kappa(o) \mid s, a) = P(f(s'), \kappa(o) \mid f(s), g(a)) \tag{3.34}$$

Shadow state constraints:

$$P(f_\xi(s'), o \mid s, \langle a\kappa(o)\rangle) = P(f_\xi(s'), o \mid f(s), g_\xi(\langle a\kappa(o)\rangle)) \tag{3.35}$$

and independence constraints:

$$P(f(s'), f_\xi(s'), o \mid s, a) = P(f(s'), \kappa(o) \mid s, a) \cdot P(f_\xi(s'), o \mid s, \langle a\kappa(o)\rangle) \tag{3.36}$$

The equations given are for the shadow model test for the model of Figure 3.5(b). Rather than redefining the compatibility test constraints, and repeating the shadow test and

Procedure 3.10.1 abstactOC-POMDP($u, \{T_{ao}\}$)

// Initialize $f : S \to \bar{S}$ so that:
$F(i,j) = \begin{cases} 1 & \text{if } f(s_i) = \bar{s}_j \\ 0 & \text{otherwise} \end{cases}$
$f(i) = f(j) \iff T_{ao}u(i) = T_{ao}u(j)$
while F has changed **do**
  for all $a \in A, o \in O$ **do**
    // construct f such that:
    $f(i) = f(j) \iff (T_{ao}F)(i,\cdot) = (T_{ao}F)(j,\cdot)$
    // update F to match f if f has changed
return F

compatibility test proofs for this model, these tests for the OC-POMDP will be defined in terms of the PSR test algorithms from Procedure 3.5.1 and Procedure 3.7.1. State mapping matrices F_χ and F_ξ or F_η should be construced using Procedure 3.10.1:

$$F_\chi \Leftarrow \text{abstractOC-POMDP}(u, \{T_{\chi(ao)}\}) \tag{3.37}$$

$$F_\xi \Leftarrow \text{abstractOC-POMDP}(e, \{T_{\xi(ao)}\}) \tag{3.38}$$

$$F_\eta \Leftarrow \text{abstractOC-POMDP}(e, \{T_{\eta(ao)}\}) \tag{3.39}$$

rather than PSR projection matrices. The Shadow Model and Shadow Compatibility test for the PSR case can be applied directly at this point, using Procedure 3.5.1 and Procedure 3.7.1, but with the state based definitions of F_χ and F_ξ or F_η.

Procedure 3.10.1 constructs the state mapping matrix F for the given parameters u and $\{T_{ao}\}$. Corresponding abstract transition matrices can be calculated according to the following rule:

$$\bar{T}_{ao}(f(i),\cdot) \stackrel{\text{def}}{=} T_{ao}F(i,\cdot). \tag{3.40}$$

These transition matrices have the following property:

$$F\bar{T}_{ao} = T_{ao}F$$

much like a PSR (Equation 3.3).

The next theorem shows that whenever the POMDP test accepts, the OC-POMDP test accepts. The proof is shown for the shadow model test, but the same theorem holds for the compatibility test.

Recall that the POMDP Shadow Model test homomorphism constraints are as follows, for every action a, observation o, state s and next state s'. Abstract state constraints:

$$\begin{aligned} P(f(s') \mid s, a) &= P(f(s') \mid f(s), g(a)) \\ P(\kappa(o) \mid s', a) &= P(\kappa(o) \mid f(s'), g(a)) \end{aligned} \tag{3.41}$$

The shadow model constraints are:

$$\begin{aligned} P(f_\xi(s') \mid s, \langle a\kappa(o)\rangle) &= P(f_\xi(s') \mid f(s), g_\xi(\langle a\kappa(o)\rangle)) \\ P(o \mid s', \langle a\kappa(o)\rangle) &= P(o \mid f_\xi(s'), g_\xi(\langle a\kappa(o)\rangle)) \end{aligned} \tag{3.42}$$

and the independence constraints are:

$$\begin{aligned} P(f(s'), f_\xi(s') \mid s, a) &= P(f(s') \mid s, a) \cdot P(f_\xi(s') \mid s, a) \\ P(f(s'), f_\xi(s') \mid b_\lambda) &= P(f(s') \mid b_\lambda) \cdot P(f_\xi(s') \mid b_\lambda). \end{aligned} \tag{3.43}$$

Theorem 3.13. *Any pair M, κ that satisfies the POMDP constraints also satisfies the OC-POMDP constraints.*

Proof. Although only the proof for the Shadow Model case is included, the results can be extended to the Shadow Compatibility test.

To derive constraint Equation 3.34 from Equation 3.41:

$$\begin{aligned} P(f(s'), \kappa(o) \mid s, a) &= \sum_{s'' \in [s']_f} P(\kappa(o) \mid s'', a) \cdot P(s'' \mid s, a) \\ &= P(\kappa(o) \mid f(s'), g(a)) \cdot P(f(s') \mid f(s), g(a)) \\ &= P(f(s'), \kappa(o) \mid f(s), g(a) \end{aligned}$$

The derivation from Equation 3.42 to Equation 3.35 is similar.

To derive Equation 3.36 from Equation 3.43:

$$\begin{aligned} P(\bar{s}', \tilde{s}', o \mid s, a) &= \sum_{s' \in \bar{s}' \cap \tilde{s}'} P(o \mid s', a) \cdot P(s' \mid s, a) \\ &= \sum_{s' \in \bar{s}' \cap \tilde{s}'} \frac{P(o \mid s', a)}{P(\kappa(o) \mid s', a)} \cdot P(\kappa(o) \mid s', a) \cdot P(s' \mid s, a) \\ &= P(o \mid \tilde{s}', g_\xi(\langle a\kappa(o) \rangle)) \cdot P(\bar{(o)} \mid \bar{s}', g_\chi(a)) \cdot \sum_{s' \in \bar{s}' \cap \tilde{s}'} P(s' \mid s, a) \\ &= P(o \mid \tilde{s}', g_\xi(\langle a\kappa(o) \rangle)) P(\tilde{s}' \mid f_\xi(s), g_\xi(a)) \cdot \\ &\quad P(\kappa(o) \mid \bar{s}', g_\chi(a)) \cdot P(\bar{s}' \mid f_\chi(s), g_\chi(a)) \\ &= P(\tilde{s}', o \mid f_\xi(s), g_\xi(\langle a\kappa(o) \rangle)) \cdot P(\bar{s}', \kappa(o) \mid f_\chi(s), g_\chi(a)) \\ &= P(\tilde{s}', o \mid s, \langle a\kappa(o) \rangle) \cdot P(\bar{s}', \kappa(o) \mid s, a) \end{aligned}$$

□

The converse is not true, however. As Table 3.3 shows, the POMDP of Figure 3.4, with the observation mapping function $\kappa(o) = o \mod 4$ provides a sample (M, κ) pair which the OC-POMDP test accepts, while the POMDP test rejects.

Lemma 3.14. *The OC-POMDP test accept set is a superset of the POMDP test accept set:* $A(POMDP) \subset A(OC - POMDP)$.

Proof. Theorem 3.13 shows that $A(POMDP) \subseteq A(OC - POMDP)$ and the POMDP example in Table 3.3 demonstrates that the sets are not equivalent. □

Domain	Observation Map (κ)	POMDP	OC-POMDP	PSR
Binary Integer Figure 3.4 with 5 bits	$\kappa(o) = o \mod 4$	fail	pass	pass

Table 3.3. Comparison of observation-directed OC-POMDP, POMDP and PSR algorithms for the POMDP of Figure 3.4.

The next theorem proves that $A(OC-POMDP) \subseteq A(PSR)$ by proving that every pair accepted by OC-POMDP is also accepted by the PSR test.

Theorem 3.15. *Any pair M, κ that satisfies the OC-POMDP constraints (Equations 3.34 - 3.36) also satisfies the PSR constraints (Equations 3.18 and 3.19) for some pair of abstract and shadow PSRs.*

Proof. This proof assumes that an abstract OC-POMDP that passes the shadow model tests is given. From this, we show that a PSR that passes the PSR shadow model test can be constructed.

Assume that F_χ and F_ξ have been calculated according to Equations 3.37 and 3.38, and that they have associated abstract transition matrix sets $\{\bar{T}_{\chi(ao)}\}$ and $\{\bar{T}_{\xi(ao)}\}$ defined according to Equation 3.40. Further, assume that these two matrices obey the Shadow Model test constraints. That is, if the joint mapping matrix F is defined from F_χ and F_ξ as in Equation 3.16, then Equations 3.18 and 3.19 are satisfied:

$$b_\lambda F = \bar{b}_{\chi(\lambda)} \otimes \tilde{b}_{\xi(\lambda)}$$
$$T_{ao} F = F \left(\bar{T}_{\chi(ao)} \otimes \bar{T}_{\xi(ao)} \right),$$

where $\bar{b}_{\chi(\lambda)} = b_\lambda F_\chi$ and $\tilde{b}_{\xi(\lambda)} = b_\lambda F_\xi$.

We can define a further mapping from the abstract states of the abstract OC-POMDP to a set of abstract PSR tests. If u is the target vector on which F_χ was built, define:

$$F'_\chi \Leftarrow krylovBasis(u, \{\bar{T}_{\chi(ao)}\})$$
$$F'_\xi \Leftarrow krylovBasis(e, \{\bar{T}_{\xi(ao)}\})$$

Define the abstract PSR transition matrices $\{\bar{T}'_{\chi(ao)}\}$ as the solutions to:

$$\bar{T}'_{\chi(ao)} F'_\chi = F'_\chi \bar{T}_{\chi(ao)}$$

and the shadow transition matrices $\{\tilde{T}'_{\xi(ao)}\}$ as the solutions to :

$$\tilde{T}'_{\xi(ao)} F'_\xi = F'_\xi T_{\xi(ao)}$$

The two matrices F_χ and F'_χ can be combined to construct a matrix mapping states in S to abstract tests in $\bar{Q}$: $F_\chi F'_\chi$. This composite matrix is the PSR mapping matrix that must be shown to satisfy the shadow model test constraints.

The two matrices F_ξ and F'_ξ can also be combined to construct a matrix mapping states in S to tests in the shadow basis $\tilde{Q}$: $F_\xi F'_\xi$. Together, the abstract and shadow projection matrices have a joint projection matrix $F(F'_\chi \otimes F'_\xi)$, and they satisfy Equations 3.18 and 3.19. With the joint projection matrix $F(F'_\chi \otimes F'_\xi)$:

$$b_\lambda(F(F'_\chi \otimes F'_\xi)) = \bar{b}'_{\chi(\lambda)} \otimes \tilde{b}'_{\xi(\lambda)}$$
$$T_{ao}(F(F'_\chi \otimes F'_\xi)) = (F(F'_\chi \otimes F'_\xi)) \left(\bar{T}'_{\chi(ao)} \otimes \bar{T}'_{\xi(ao)}\right)$$

The initial belief state constraint is straightforward to verify. For the transition constraint:

$$\begin{aligned} T_{ao} F(F'_\chi \otimes F'_\xi) &= F\left(\bar{T}_{\chi(ao)} \otimes \bar{T}_{\xi(ao)}\right)(F'_\chi \otimes F'_\xi) \\ &= F\left(\bar{T}_{\chi(ao)} F'_\chi \otimes \bar{T}_{\xi(ao)} F'_\xi\right) \\ &= F\left(F'_\chi \bar{T}'_{\chi(ao)} \otimes F'_\xi \bar{T}'_{\xi(ao)}\right) \\ &= F(F'_\chi \otimes F'_\xi)\left(\bar{T}'_{\chi(ao)} \otimes \bar{T}'_{\xi(ao)}\right) \end{aligned}$$

Procedure 3.11.1 observationMapCompatibiliy(M, κ)

```
// Construct the abstract PSR
F_χ ⇐ KrylovBasis(u, {T_χ(ao)})

// Construct the shadow PSR
F_η ⇐ KrylovBasis(e, {T_η(ao)})

I_c ⇐ psrCompatibilityMatrix(M, F_χ, F_η)
U ⇐ an empty set of state pairs (implemented as a queue)
∼_κ: S × S → { true, false}
initialize ∼_κ so that κ(o_i) = κ(o_j) ⟺ o_i ∼_κ o_j

∼_κ⇐ observationConditionalSplits(M, κ, ∼_κ) // Procedure 3.11.2
makeDistributionsCompatible(M, b_λ, b_λ, I_c, U) // Procedure ??
while U not empty do
   ⟨i, j⟩ ⇐ an element removed from U
   ∼_κ⇐ makeObservationsCompatible(M, i, j, ∼_κ) // Procedure 3.11.3
   makeStateDistributionsCompatible(i, j, I_c, U) // Procedure 3.11.4
   // mark the two states as compatible
   I_c(⟨i, j⟩, ⟨i, j⟩) ⇐ 1
return κ
```

and therefore the two matrices $F_\chi F'_\chi$ and $F_\xi F'_\xi$ with core tests $\bar{Q}$ and $\tilde{Q}$ satisfy the PSR shadow model test requirements, and the pair (M, κ) would be accepted under the PSR constraints. □

3.11 Observation Splitting

Thus far this chapter has focused on the PSR and OC-POMDP acceptance tests. This section outlines a corresponding abstract model search algorithm.

If Procedure 3.7.1 fails, it must be the case that Procedure 3.7.3 failed for the initial belief state. If all states were compatible ($I_c =$ an identity matrix), Procedure 3.7.3 would succeed, therefore, there must be some set U of pairs of states where if every pair in U were compatible, Procedure 3.7.3 would succeed. Procedure 3.11.1 proceeds in the same manner as Procedure 2.7.2. Starting with the initial belief state, it generates a list of state pairs that must be compatible to make Procedure 3.7.3 succeed, under the current F_χ and

Procedure 3.11.2 observationConditionalSplits($M, \kappa, \sim_\kappa$)

for all $s_k \in S, a \in A$ **do**
  for all $\bar{o} \in \bar{O}$ **do**
    for all $o_i, o_j \in \bar{o}$ **do**
      if $(P(o_i \mid s_k, a) > 0) \wedge (P(o_j \mid s_k, a) > 0)$ **then**
        if $T_{ao_i} F_\chi(k, \cdot) \cdot T_{ao_j} u^T(k) \neq T_{ao_j} F_\chi(k, \cdot) \cdot T_{ao_i} u^T(k)$ **then**
          $o_i \sim_\kappa o_j \Leftarrow$ **false**
        if $T_{\xi(ao_i)} F_\eta(k, \cdot) \cdot T_{\xi(ao_j)} e^T(k) \neq T_{\xi(ao_i)} F_\eta(k, \cdot) \cdot T_{\xi(ao_j)} e^T(k)$ **then**
          $o_i \sim_\kappa o_j \Leftarrow$ **false**
return $\sim_\kappa$

Procedure 3.11.3 makeObservationsCompatible($M, i, j, \sim_\kappa$)

for all $s_i, s_j \in S$ **do**
  for all $a \in A$ **do**
    if $T_{ao_k} u^T(i) \cdot T_{\xi(ao_l)} e^T(j) \neq T_{ao_k} u^T(i) \cdot T_{\xi(ao_l)} e^T(j)$ **then**
      $o_i \sim_\kappa o_j \Leftarrow$ **false**
return $\sim_\kappa$

F_η matrices. These changes require certain changes in the observation mapping function. As with Procedure 2.7.2, there may be multiple observation mapping functions that produce the desired changes in the state compatibility function. Procedure 3.11.1 therefore constructs an observation compatibility relation $\sim_\kappa$: $O \times O \rightarrow \{true, false\}$ rather than an equivalence relation over observations that could be used to construct κ directly. Any grouping of observations into abstract observations that does not group any incompatible observation pairs is acceptable.

Compatibility of any pair of states s_i adn s_j is determined by the ability to construct the row $W_{ao}(\langle i, j \rangle, \cdot)$ for every action a and observation o. It is important to be able to determine why the construction of this vector failed for a particular row $\langle i, j \rangle$. There are two possible reasons:

1. Incompatible state pairs in I_c. If this is the cause, then setting $I_c = I$ will produce a solution. Procedure 3.11.4 addresses this issue, by adding state pairs to the merge list U.

2. Inability to solve both Equation 3.25 and Equation 3.26 using the same solution. Procedures 3.11.1 and 3.11.2 address this issue by marking observation pairs incompatible.

In order to correct the observation mapping function, the factors due to the immediate observations (item 2) must be separated from the factors due to the next state distribution (item 1). This section focuses on developing a set of observation constraints, such that if these constraints are satisfied, and $I_c = I$, then it must be possible to solve for $W_{ao}(\langle i, j \rangle, \cdot)$ for states s_i and s_j.

Procedure 3.11.2 implements the first separate piece of the observation constraints. As a constraint over states, this method seeks to find an observation compatibility function that will satisfy the following rule. Any state s_k has consistent next state predictions if, for all actions a and observations o:

$$T_{\chi(ao)} F_\chi(k, \cdot) \cdot T_{ao} u^T(k) = T_{ao} F_\chi(k, \cdot) \cdot T_{\chi(ao)} u^T(k) \tag{3.44}$$

$$T_{\eta(ao)} F_\eta(k, \cdot) \cdot T_{\xi(ao)} e^T(k) = T_{\xi(ao)} F_\eta(k, \cdot) \cdot T_{\eta(ao)} e^T(k) \tag{3.45}$$

In the observation improvement method, all states must have consistent next state predictions. Turning this into a constraint on the binary observation compatibility relation, two observations should only be compatible $o_l \sim_\kappa o_k$ if for all actions a and states s_k:

$$T_{ao_i} F_\chi(k, \cdot) \cdot T_{ao_j} u^T(k) = T_{ao_j} F_\chi(k, \cdot) \cdot T_{ao_i} u^T(k)$$

$$T_{\xi(ao_i)} F_\eta(k, \cdot) \cdot T_{\xi(ao_j)} e^T(k) = T_{\xi(ao_i)} F_\eta(k, \cdot) \cdot T_{\xi(ao_j)} e^T(k)$$

If the new observation mapping function κ conforms to the observation compatibility function, then summing over $o_i \in \kappa(o_j)$ on both sides of each equation yields the desired constraints, Equations 3.44 and 3.45, for all states and actions. This constraint on the observation compatibility function is implemented in Procedure 3.11.2.

In the PSR case, Equation 3.44 can be reduced to:

$$P(\bar{q}, o \mid s, a) = P(\bar{q}, \kappa(o) \mid s, a)$$

and Equation 3.45 becomes:

$$P(\bar{q}_\eta, o \mid s_j, \langle a, \kappa(o)\rangle) = P(\bar{q}_\eta, \kappa(o) \mid s_j, \langle a, \kappa(o)\rangle)$$

for every s in S, a in A, $\bar{q}$ in $\bar{Q}$ and $\bar{q}_\eta \in \bar{Q}_\eta$. This constraint amounts to a requirement that the abstract next state vector not depend on the observation label, if the abstract observation label is given.

The next portion of the observation split criteria more closely resembles Equation 2.54, from the previous chapter. Two states s_i and s_j can only be compatible if their observation ratios are compatible at the next time step, for every action a and observation o:

$$T_{ao}u^T(i) \cdot T_{\eta(ao)}e^T(j) = T_{\chi(ao)}u^T(i) \cdot T_{\xi(ao)}e^T(j) \tag{3.46}$$

In the observation splitting algorithm, if two states s_i and s_j are required to become compatible, the observation compatibility function should be constructed such that two observations are only compatible ($o_l \sim_\kappa o_k$) if:

$$T_{ao_l}u^T(i) \cdot T_{\xi(ao_k)}e^T(j) = T_{ao_k}u^T(i) \cdot T_{\xi(ao_l)}e^T(j)$$

this implies that if κ respects the observation compatibility constraints, then summing over $o_l \in \kappa(o_k)$ yields the desired constraint, Equation 3.46. In the specific case of an observation-directed abstract PSR, in which $u = e$, two states s_i and s_j are observa-

Procedure 3.11.4 makeStateDistributionsCompatible(i, j, I_c, U)

// Abstract projection matrix (from Equation 3.25)
$abstractProjection \Leftarrow I(F_\chi \otimes I)$
// Availability projection matrix (from Equation 3.26)
$availabilityProjection \Leftarrow I(I \otimes F_\eta)$
$A \Leftarrow [abstractProjection : availabilityProjection]$

for all $a \in A, o \in O$ **do**
  // Abstract prediction matrix (from Equation 3.25)
  $abstractPrediction \Leftarrow (T_{ao} \otimes T_{ao})(F_\chi \otimes I)$
  // Availability prediction matrix (from Equation 3.26)
  $availabilityPrediction \Leftarrow (T_{ao} \otimes T_{ao})(I \otimes F_\eta)$
  $Y \Leftarrow [abstractPrediction : availabilityPrediction]$

  // Solve the system of equations for the $\langle i, j \rangle^{th}$ row of W_{ao}
  $y \Leftarrow Y(\langle i, j \rangle, \cdot)$ // The $\langle i, j \rangle^{th}$ row
  // Given Procedures 3.11.2 and 3.11.3, the solution x to $A^T x = y^T$ should exist.
  $W_{ao}(\langle i, j \rangle, \cdot) = x$
  for all $k, l \in S$ **do**
    if $x(\langle k, l \rangle) > 0$ **then**
      $U \Leftarrow U \cup \{\langle k, l \rangle)$

tion ratio compatible (Equation 3.46) if, for all observations o and actions a, whenever $\eta(\kappa(o), s_j, a) \neq 0$:

$$\frac{P(o \mid s_i, a)}{P(\kappa(o) \mid s_i, a)} = \sum_{s' \in S} \frac{P(o \mid s', a)}{P(\kappa(o) \mid s', a)} \cdot P(s' \mid s_j, a)$$

The reward-directed case differs slightly. Two states s_i and s_j are observation compatible (Equation 3.46) if, for all observations o and actions a:

$$\frac{E(r \mid s_i, a, o) \cdot P(o \mid s_i, a)}{E(r \mid s_i, a, \kappa(o)) \cdot P(\kappa(o) \mid s_i, a)} = \sum_{s' \in S} \frac{P(o \mid s', a)}{P(\kappa(o) \mid s', a)} \cdot P(s' \mid s_j, a)$$

Equations 3.44, 3.45 and 3.46 are the constraints that directly concern the observation function. If these constraints are satisfied, then there is a solution for $W_{ao}(\langle i, j \rangle, \cdot)$ that satisfies both Equation 3.25 and 3.26:

Procedure 3.11.5 fixInitialBelief($b_\lambda, I_c, F_\chi, F_\eta, U$)

$A \Leftarrow [I(I \otimes F_\eta) : I_c(F_\chi \otimes I)]$
$y \Leftarrow [b_\lambda \otimes \bar{b}_{\eta(\lambda)} : b_{\chi(\lambda)} \otimes b_{\xi(\lambda)}]$
Solve for $xA = y$
for all $i, j \in S$ **do**
  if $x(\langle i, j \rangle) > 0$ **then**
    $U \Leftarrow U \cup \langle i, j \rangle$

$$xI(I \otimes F_\eta)(\langle i, j \rangle, \cdot) = (T_{ao} \otimes T_{\eta(ao)})(I \otimes F_\eta)(\langle i, j \rangle, \cdot)$$

$$xI(F_\chi \otimes I)(\langle i, j \rangle, \cdot) = (T_{\chi(ao)} \otimes T_{\xi(ao)})(F_\chi \otimes I)(\langle i, j \rangle, \cdot)$$

Theorem 3.16. *If all states are compatible ($I_c = I$), and Equations 3.44, 3.45 and 3.46 are satisfied, then the vector $c_r(T_{ao} \otimes T_{\xi(ao)})(\langle i, j \rangle, \cdot)$ where:*

$$c_r = \frac{T_{\chi(ao)} u^T(i)}{T_{ao} u^T(i)}$$

is a solution for $W_{ao}(\langle i, j \rangle, \cdot)$.

Proof. By Equation 3.46:

$$\begin{aligned} c_r &= \frac{T_{\chi(ao)} u^T(i)}{T_{ao} u^T(i)} \\ &= \frac{T_{\eta(ao)} e^T(j)}{T_{\xi(ao)} e^T(j)} \end{aligned}$$

By Equation 3.44:

$$\begin{aligned} T_{\chi(ao)} F_\chi(i, \cdot) \cdot T_{ao} u^T(i) &= T_{ao} F_\chi(i, \cdot) \cdot T_{\chi(ao)} u^T(i) \\ T_{\chi(ao)} F_\chi(i, \cdot) &= T_{ao} F_\chi(i, \cdot) \cdot \frac{T_{\chi(ao)} u^T(i)}{T_{ao} u^T(i)} \\ &= T_{ao} F_\chi(i, \cdot) \cdot c_r \end{aligned}$$

By Equation 3.45:

$$\begin{aligned} T_{\eta(ao)}F_{\eta}(j,\cdot)\cdot T_{\xi(ao)}e^{T}(k) &= T_{\xi(ao)}F_{\eta}(j,\cdot)\cdot T_{\eta(ao)}e^{T}(k) \\ T_{\eta(ao)}F_{\eta}(j,\cdot) &= T_{\xi(ao)}F_{\eta}(j,\cdot)\cdot \frac{T_{\eta(ao)}e^{T}(k)}{T_{\xi(ao)}e^{T}(k)} \\ &= T_{\xi(ao)}F_{\eta}(j,\cdot)\cdot c_r \end{aligned}$$

Therefore:

$$\begin{aligned} T_{ao}F_{\chi}(i,\cdot)\cdot c_r \otimes T_{\xi(ao)}(j,\cdot) &= T_{\chi(ao)}F_{\chi}(i,\cdot)\otimes T_{\xi(ao)}(j,\cdot) \\ c_r\cdot(T_{ao}\otimes T_{\xi(ao)})(F_{\chi}\otimes I)(\langle i,j\rangle,\cdot) &= (T_{\chi(ao)}\otimes T_{\xi(ao)})(F_{\chi}\otimes I)(\langle i,j\rangle, \\ W_{ao}(F_{\chi}\otimes I)(\langle i,j\rangle,\cdot) &= (T_{\chi(ao)}\otimes T_{\xi(ao)})(F_{\chi}\otimes I)(\langle i,j\rangle,\cdot) \end{aligned}$$

and

$$\begin{aligned} T_{ao}(i,\cdot)\otimes T_{\xi(ao)}F_{\eta}(j,\cdot)\cdot c_r &= T_{ao}(i,\cdot)\otimes T_{\eta(ao)}F_{\eta}(j,\cdot) \\ c_r\cdot(T_{ao}\otimes T_{\xi(ao)})(I\otimes F_{\eta})(\langle i,j\rangle,\cdot) &= (T_{ao}\otimes T_{\eta(ao)})(I\otimes F_{\eta})(\langle i,j\rangle,\cdot) \\ W_{ao}(I\otimes F_{\eta})(\langle i,j\rangle,\cdot) &= (T_{ao}\otimes T_{\eta(ao)})(I\otimes F_{\eta})(\langle i,j\rangle,\cdot) \end{aligned}$$

Therefore Equations 3.25 and 3.26 have at least one solution ($c_r(T_{ao}\otimes T_{\xi(ao)})(\langle i,j\rangle,\cdot)$) in common for row $\langle i,j\rangle$ under these conditions. □

Similarly, there is at least one solution for Equations 3.27 and 3.28, if all states are assumed to be compatible. This solution is $b_{\lambda}\otimes b_{\lambda}$. However, $b_{\lambda}\otimes b_{\lambda}$ and $c_r\cdot(T_{ao}\otimes T_{\xi(ao)})$ are not necessarily the optimal solutions, in terms of producing a more compact observation function or model.

The problem with this approach is that this algorithm does not prioritize links between states that are already compatible, as the graph flow matching algorithm did (Procedure

2.7.4). In the POMDP of Figure 2.8, using $b_\lambda \otimes b_\lambda$ as the initial belief distribution matching matrix while searching for the abstract model in Figure 2.8(b) results in an observation compatibility relation in which no two observations are compatible. The graph flow algorithm, when initialized using links between already compatible state pairs, produces the observation compatibility relation illustrated in Figure 2.15.

3.11.1 Graph Based Matching algorithm

Rather than working directly with the transition matrices, the graph-flow algorithm in Procedure 2.7.2 can be adapted for use with the OC-POMDP or PSR observation map acceptance constraints. It requires a state to abstract state mapping function, however, this can be constructed from the abstract and availability PSR matrices. If the OC-POMDP acceptance constraints are being used, f and f_η should be defined as in Equations 3.34 and 3.35.

If the PSR acceptance constraints are being used, define a state mapping function $f : S \to \bar{S}_\chi$ according to the abstract PSR projection matrix:

$$f(s_i) = f(s_j) \iff F_\chi(i,\cdot) = F_\chi(j,\cdot)$$

Also define a state mapping function $f_\eta : S \to \bar{S}_\eta$ according to the abstract availabiltyPSR projection matrix:

$$f_\eta(s_i) = f_\eta(s_j) \iff F_\eta(i,\cdot) = F_\eta(j,\cdot)$$

Substituting the state mapping function for F_χ and F_η may restrict the set of observation maps, however, the graph flow matching algorithm has the advantage of being faster and more accurate in general, so in many cases this trade-off may be worth it.

The graph matching algorithm remains largely unchanged from Procedure 2.7.2. Procedures 3.11.2 and 3.11.3 must be applied to ensure that Equations 3.44, 3.45 and 3.46 are satisfied before the graph matching step may be performed.

For each state pair (i, j) to be merged, rather than one graph matching per action a, there must be one set of matching graphs for each matrix T_{ao}. The vertices of the flow graph for $\bar{s}_l$ and $\bar{s}_r$ are:

- s (source node)
- $L = \{l_m \mid m \in \bar{s}_l\}$ (state nodes in $\bar{s}_l$)
- t (sink node)
- $R = \{r_n \mid n \in \bar{s}_r\}$ (state nodes in $\bar{s}_r$)

and edge capacities:

- $cap(s, l_m) = \frac{P(s_m, o|s_i, a)}{P(f(s_m), \kappa(o)|s_i, a)}$
- $cap(l_n, r_m) = \begin{cases} 1 & \text{if } s_n \sim_c s_m \\ 0 & \text{otherwise} \end{cases}$
- $cap(r_n, t) = \frac{P(s_n, o|s_j, \langle a\kappa(o)\rangle)}{P(f_\eta(s_n), \kappa(o)|s_j, \langle a\kappa(o)\rangle)}$

when calculating the compatibility matrix initially, and $cap(l_n, r_m) = 1$ in the augmented graph when merging states i and j.

The graph flow algorithm for these graphs is unchanged. Equations 3.44, 3.45 and 3.46 ensure that when all states are assumed to be compatible ($cap(l_n, r_m) = 1$), this graph flow problem has a solution.

Theorem 3.17. *If Equations 3.44, 3.45 and 3.46 are satisfied for states i and j, then a flow matching exists. This is due to the fact that the total outgoing capacity for the source, and the total incoming capacity for the sink are equivalent.*

Proof. By Equations 3.44, 3.45 and 3.46, for any pair $\bar{s}_l \in \bar{S}$ and $\bar{s}_r \in \bar{S}_\eta$:

$$\begin{aligned}
\sum_{s_m \in \bar{s}_r} cap(s, l_m) &= \sum_{s_m \in \bar{s}_r} \frac{P(s_m, o \mid s_i, a)}{P(f(s_m), \kappa(o) \mid s_i, a)} \\
&= \frac{P(o \mid s_i, a)}{P(\kappa(o) \mid s_i, a)} \\
&= \frac{P(o \mid s_j, \langle a\kappa(o) \rangle)}{P(\kappa(o) \mid s_j, \langle a\kappa(o) \rangle)} \\
&= \sum_{s_n \in \bar{s}_l} \frac{P(s_n, o \mid s_j, \langle a\kappa(o) \rangle)}{P(f_\eta(s_n), \kappa(o) \mid s_j, \langle a\kappa(o) \rangle)} \\
&= \sum_{s_n \in \bar{s}_l} cap(r_n, t)
\end{aligned}$$

Since the graph between the source and the sink is fully connected when all states are assumed to be compatible, the graph flow problem has a solution. □

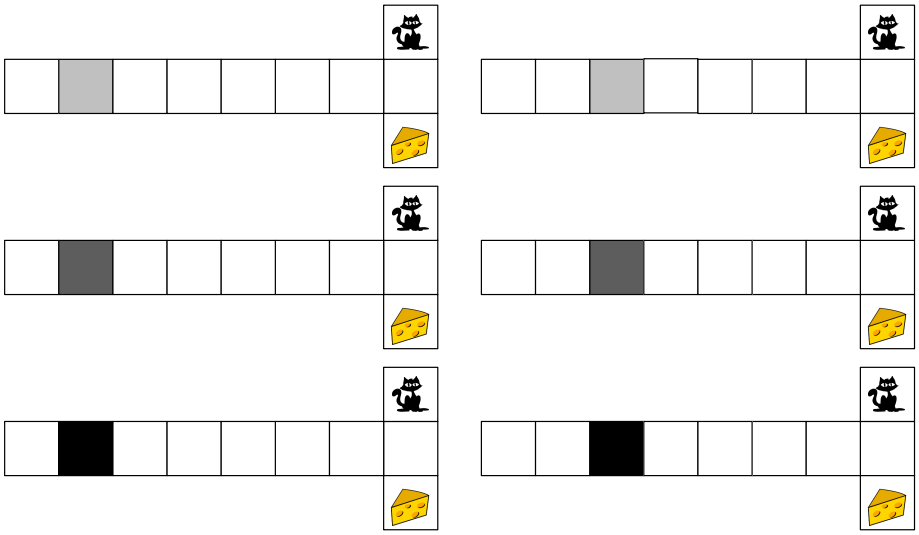
3.12 Time Experiments: Comparison to Existing Work

This section compares the performance of the OC-POMDP acceptance criteria, with the graph flow search algorithm, to an existing history-based abstraction search algorithms by Talvitie et al. (2008). This existing algorithm can be applied to an existing model. In this form, it:

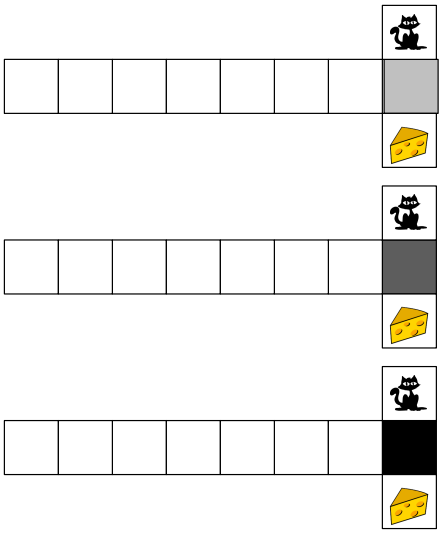
- takes as input a pair (M, κ) where M is a PSR and κ is an observation mapping function
- generates an observation compatibility function $\sim_\kappa$: $O \times O \rightarrow \{true, false\}$ as output

It solves the same problem as a single execution of Procedure 2.7.2. These experiments therefore compare a single application of Procedure 2.7.2 to the performance of the history-based algorithm.

The history-based algorithm examines pairs of histories. If two histories have different abstract predictions, then any history mapping function χ that gives them the same label



(a) (b)



(c)

Figure 3.6. Hallway domains in which the distance to the distinct states varies.

must be invalid. Talvitie et al. (2008) use two useful results to search over the set of history pair for observation incompatibilities:

- only history pairs that differ by one observation (and no actions) must be compared. If the outcomes of the two histories differ, these two differing observations are not compatible.

- only history pairs of length less than or equal to the size of Q, the core tests for the original system, need be examined.

The maximum number of history pairs of length n is $(|A| \cdot |O|^n)$, so, while the result is useful in that it shows that a finite number of history pairs need to be examined, the number of histories examined may still be exponential in the size of Q. They demonstrate that in some cases, the correct observation compatibility relation is constructed at a much shorter history length, however.

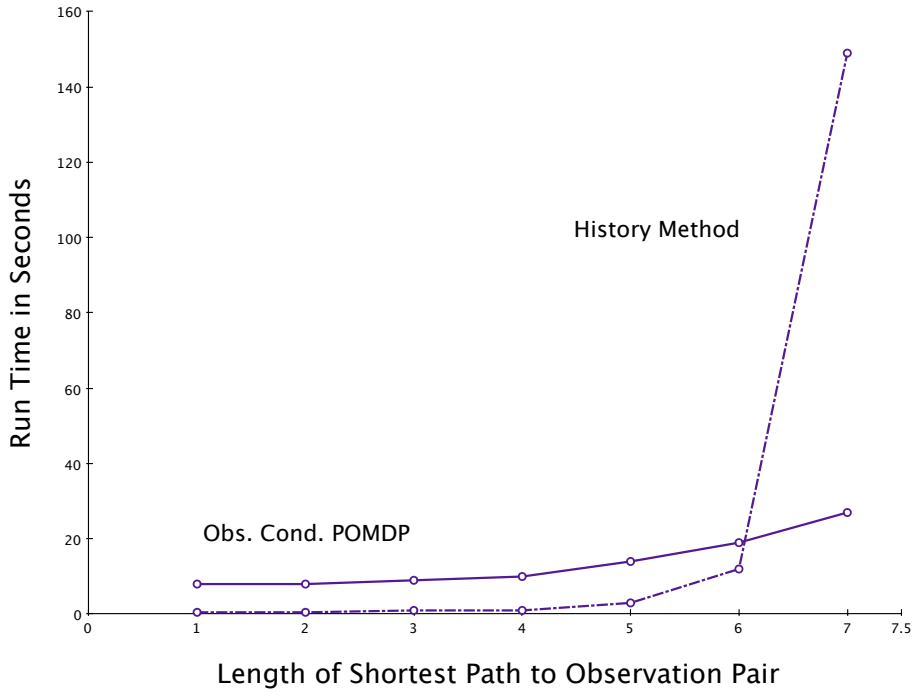


Figure 3.7. Comparison of the History Method and OC-POMDP method.

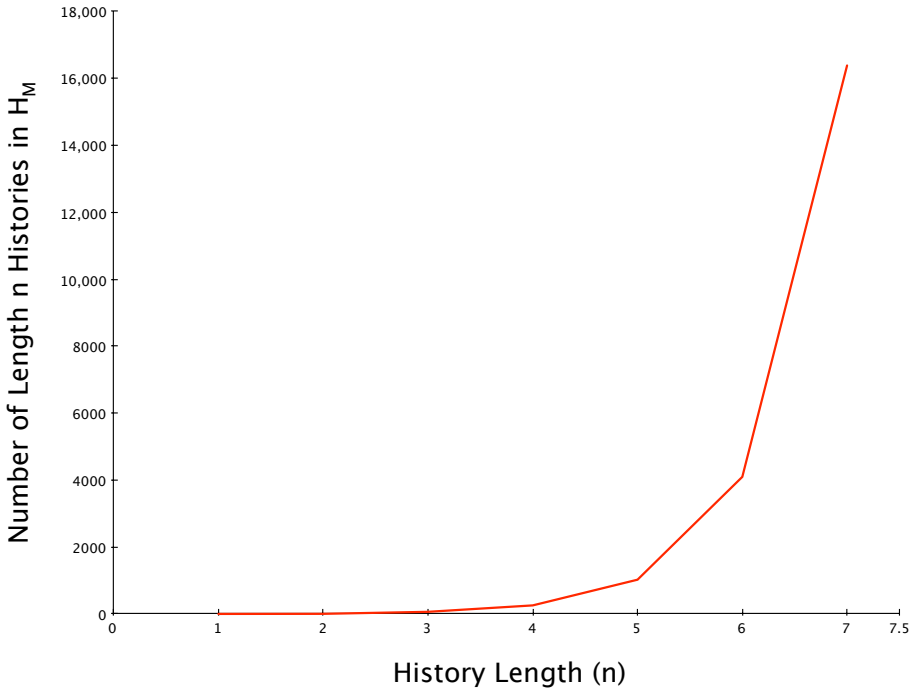


Figure 3.8. The number of histories of length n for n from 1 to 7.

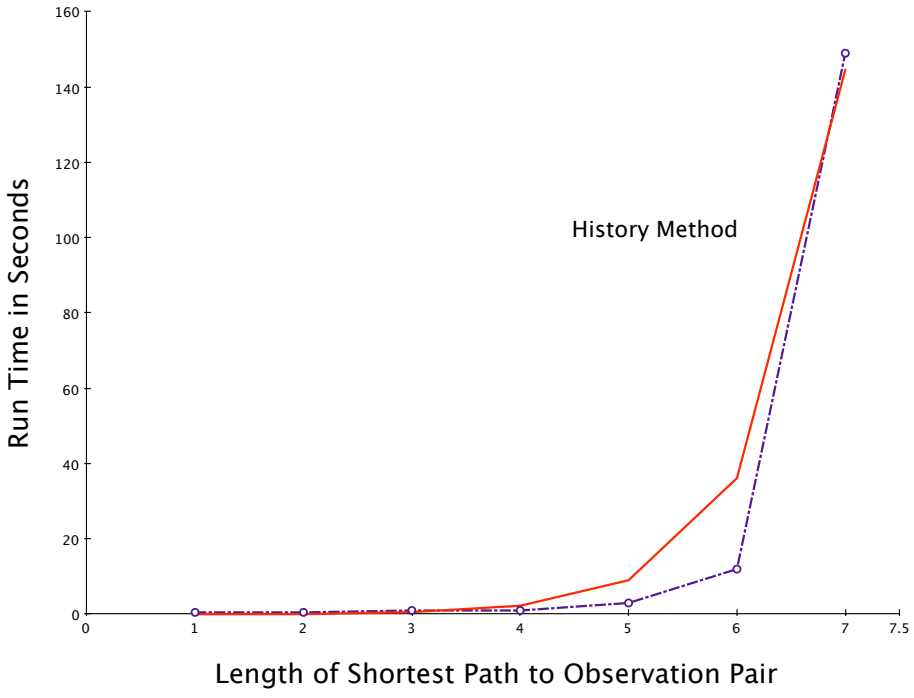


Figure 3.9. Comparison of the time and history length curves for the history based algorithm.

Figure 3.7 shows the run time of the OC-POMDP observation splitting algorithm vs the history based observation splitting algorithm. These results were gathered from domains like the one shown in Figures 3.6(a), 3.6(b) and 3.6(c). In these experiments, 7 POMDPs were constructed. The first POMDP has the distinct observation states located one time step from the starting states in each hall (Figure 3.6(a)). The second POMDP has the distinct observation states located two steps from the starting states (Figure 3.6(b)), etc. The final, seventh POMDP (Figure 3.6(c)) has distinct observations at 7 steps from the starting states. The initial observation mapping function κ given to both algorithms was:

$$\bar{o}_0 = \{cheese\}\bar{o}_1 \qquad = \{cat\}\bar{o}_2 = \{white, lightgrey, grey, black\}$$

And the observation compatibility function $\sim_\kappa$ returned by both algorithms was the one illustrated in Figure 2.15.

In these domains the determining factor in how long the histories must be in the history-based algorithm of Talvitie et al. (2008) is the distance between the starting states and the three states with distinct observations. The history method does well when the observation distinction is close to the initial states (the left of the hallways), but eventually the growth in the number of histories to examine causes the run time to increase far above the run time of the OC-POMDP algorithm.

In these experiments, the history method was run to a fixed history length of 1 for the first POMDP, 2 for the second, and so on. Figure 3.8 is a graph of the number of histories of length n, for $n = 1$ to 7. In Figure 3.9, this curve is normalized to fit the range of the curve of running times, demonstrating that the shape of the two curves is similar. In general, the depth of search needed would not be known, and the algorithm could not verify that the observation compatibility function is complete at the point at which it halts in these experiments.

The OC-POMDP algorithm, on the other hand, was run to completion, and verifies that the observation compatibility function it finds is complete. The number of states pairs

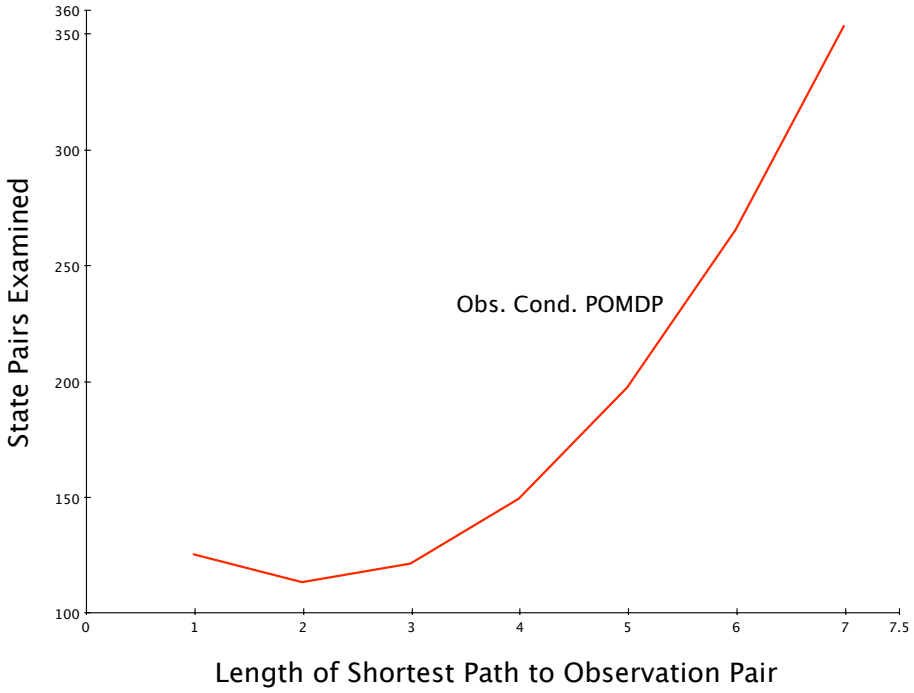


Figure 3.10. The number of state pairs examined for each POMDP, from 1 to 7 states between the initial belief and the state distinctions.

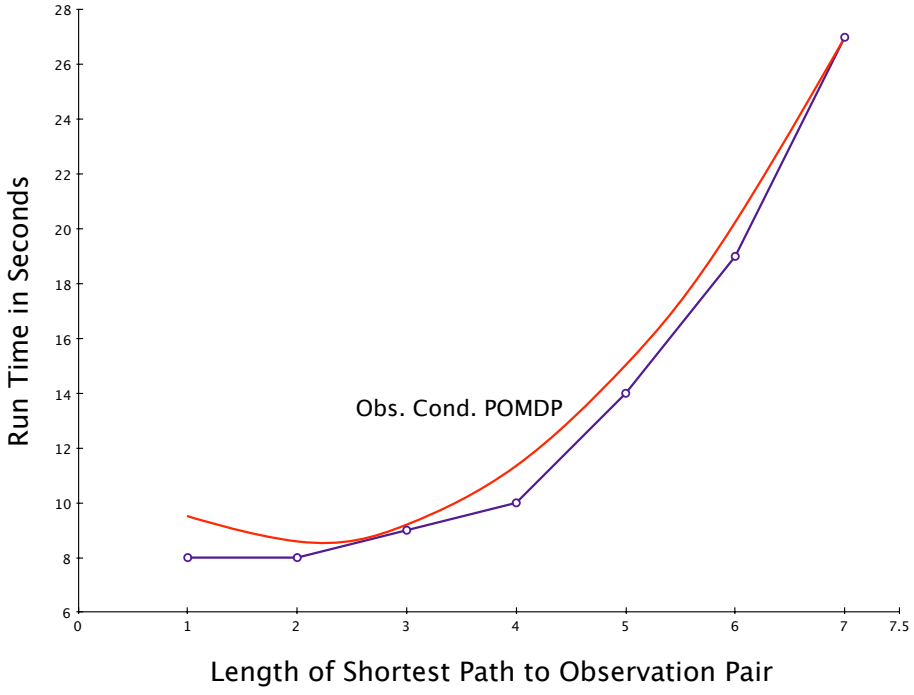


Figure 3.11. Comparison of the shape of the curve representing the number of state pairs examined by OC-POMDP, and the number of second to completion of the algorithm.

the algorithm must merge in order to find the desired observation distinction increases automatically as the observation distinction moves further from the start of the hallways. Figure 3.10 is a graph of the number of state pair merges performed by the algorithm for each domain, from 1 to 7. Figure 3.11 fits this curve to the range of run times for the algorithm, and shows that the run time and number of state pairs examined have similar curves as the domain complexity increases.

3.13 Conclusion

This chapter developed algorithms that construct the abstract shadow model and abstract availability models as PSRs. This adds the option of constructing reward-directed abstract models, which may be more compact than output-directed models for the same target function. When the abstract PSR model is output-directed, the observation abstractions accepted by the shadow model and compatibility tests have been shown to be better than those accepted by the abstract POMDP approach covered in Chapter 2. In some cases, this is due not to the fact that the abstract PSR is built using basis vectors, but due to the structure of the PSR model. In fact, a reasonable change to the abstract POMDP structure to an OC-POMDP structure can remedy this difference.

CHAPTER 4

CONCLUSION

This dissertation focused on the case where the objective is to form an abstract model based on a specific output function. Some examples of output functions include features of objects, like size, position, color, etc. as well as features like “Is it raining?” or “Am I tired?”. This type of model strikes a balance between abstract model compactness and re-usability. Output function based abstract models can be re-used for families of tasks based on their output function (“Move object to location x”), but they are not general purpose models. Tasks where the definition of the goal of the task depends on other variables are outside the scope of an output function specific model. So, for example, while the model for agent location can be used for general navigation tasks, it cannot be used, for example, to learn how to open a jar.

This tension between model size and re-usability may become more of an issue as agents become less specialized. It may be practical currently to build a fixed abstraction into the agent’s internal structure, however, this will not be practical for more general purpose agents. A general purpose warehouse loading agent should be capable of adapting to new materials in the warehouse, a general purpose housekeeping agent should be able to adapt to new tasks as objects are added to its environment and need to be cleaned. However, this does not necessarily mean that the agents must have a fully general purpose model of their entire environment. In these examples the tasks for the agents were drawn from a family of related tasks.

One of the main open questions this type of framework raises is the problem of choosing a good set of output functions for an agent. Take the example of a mail delivery agent in

an office building. The same agent might construct many smaller abstract models, one for each destination it must reach, or it can build a more general purpose "agent location" abstraction for navigation. The overall learning efficiency of the agent over its lifetime will depend on the time needed to construct the abstract models, the time needed to plan or learn policies for each task in the abstract models, and the number of times each abstract model is re-used.

This dissertation addressed a more basic problem, however. Agents such as the ones described briefly above would occupy complex domains, where the state is not fully observable. Finding appropriate output function specific abstractions under these conditions is quite difficult. Wingate et al. (2007) have demonstrated that when the state is relational, consisting of objects and their relations to each other, abstractions like this can be useful. However, they hand-craft the abstract models, and do not therefore include the time needed to construct the abstract model in their calculations. Talvitie et al. (2008) provide a worst case exponential time algorithm for finding such abstractions under these conditions.

This dissertation defined several alternative polynomial time search algorithms for finding output function based abstractions. These algorithms address the idealized in that:

- they accept only perfectly accurate models
- they must be provided with an accurate original model.

However, they also relax the search problem in two ways:

- they sometimes reject accurate models in favor of larger abstractions
- even with the set of acceptable abstract models, the search strategy may not find the smallest possible abstract model.

These approximations allow the algorithms to operate by examining local characteristics of the original model. For example, the shadow model tests examine the abstract next state distributions for individual (state, action) pairs. This avoids the problem of examining the

properties of either histories or belief state vectors directly, as the sets of histories and belief state vectors can both be quite large.

However, this does not change the fact that these algorithms are designed for an ideal that rarely exists. Take the example of the chess player, concentrating on the board and ignoring the surroundings. While it would be hard to detect the affect that a nearby pigeon has on the game, it is possible that there are some small details (the expression of a spectator, observing the board or the way that the weather affects the mood of the opponent) that would assist the player in making predictions in some small way. Approximate algorithms that take this into account, ranking the observation distinctions by the amount to which they affect predictions, can handle this case more appropriately.

Approximate algorithms can in many cases also perform faster than the idealized algorithms outlined here. An approximate solution to the linear equations in Procedure 3.7.2 or the graph flow algorithm of Procedure 2.7.2 may produce a model that has good performance more quickly.

Technically, the hardest part of the algorithms outlined here is the choice between multiple observation mapping functions in the observation map improvement step of Procedure 2.7.6. A measure of the cost of distinguishing between particular pairs of observations in terms of the increase in the complexity of the abstract model would lead to better heuristics for this step.

The algorithms outlined here solve an important problem: finding abstract models for specific output functions when the state is partially observable. They should serve only as a starting point for developing more practical approximate algorithms, however.

BIBLIOGRAPHY

Craig Boutilier, Ray Reiter, and Bob Price. Symbolic dynamic programming for first-order mdps. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 690–697, 2001.

Michael Bowling, Ali Ghodsi, and Dana Wilkinson. Action respecting embedding. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, pages 65–72, 2005.

A. Carlin and S. Zilberstein. Value-based observation compression for dec-pomdps. In *Proceedings of the Seventh International Conference on Autonomous Systems and Multiagent Systems (AAMAS)*, pages 501–508, 2008.

Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2009.

Thomas Dean and Robert Givan. Model minimization in markov decision processes. In *Proceedings of AAAI*, 1997.

Richard Dearden and Craig Boutilier. Abstraction and approximate decision theoretic planning. *Artificial Intelligence*, 89(1):219–283, 1997.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Journal of Artificial Intelligence Research*, 2003.

J. Hartmanis and R. E. Stearns. *Algebraic Structure Theory of Sequential Machines*. Prentice-Hall, Englewood Cliffs, N.J., 1966.

Michael P. Holmes and Charles Lee Isbell, Jr. Looping suffix tree-based inference of partially observable hidden state. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998.

J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. D. Van Nostrand, New York, 1960.

Michael L. Littman, Richard S. Sutton, and Satinder P. Singh. Predictive representations of state. In *Advances In Neural Information Processing Systems*, volume 14, 2001.

Sridhar Mahadevan. Samuel meets amarel: Automating value function approximation using global state space analysis. In *Proceedings of the 20th National Conference on Artificial Intelligence*, 2005.

Andrew K. McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, Rochester University, 1995.

Ann Nicholson and Leslie Pack Kaelbling. Toward approximate planning in very large stochastic domains. In *Proceedings of the AAAI Spring Symposium on Decision Theoretic Planning*, Stanford, CA, 1994. URL `citeseer.ist.psu.edu/nicholson94toward.html`.

D.M. Park. Concurrency on automata and infinite sequences. In P. Deussen, editor, *Conference on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*. Springer Verlag, 1981.

Avi Pfeffer. Sufficiency, separability and temporal probabilistic models. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 421–428, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-800-1.

Pascal Poupart and Craig Boutilier. Value-directed compression of pomdps. In *Advances in Neural Information Processing Systems 15 (NIPS)*, pages 1547 –1554, Vancouver, BC, 2002.

B Ravindran. *An Algebraic Approach to Abstraction in Reinforcement Learning*. PhD thesis, University of Massachusetts, 2004.

Yousef Saad. Iterative methods for sparse linear systems. *SIAM, 2nd Edition*, 2003.

Vishal Soni and Satinder Singh. Abstraction in predictive state representations. In *Proceedings of the 22nd Conference on Artificial Intelligence*, 2007.

Richard Sutton and Andrew G. Barto. *Reinforcement Learning*. MIT Press, 1998.

Erik Talvitie, Britton Wolfe, and Satinder Singh. Building incomplete but accurate models. In *Proceedings of ISAIM*, 2008.

David Wingate, Vishal Soni, Britton Wolfe, and Satinder Singh. Relational knowledge with predictive state representations. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007.

Alicia Peregrin Wolfe and Andrew G. Barto. Decision tree methods for finding reuseable mdp homomorphisms. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.